Energy efficient supplemental lighting controls in greenhouses

Schuyler W Duffy

Cornell University

August 2019

BIOGRAPHICAL SKETCH

Schuyler Duffy is a horticultural engineer specializing in translating evidence-based research into tractable algorithms for controlled environment agriculture. Before studying at Cornell University, Schuyler received a BA from University of California, Los Angeles.

ACKNOWLEDGEMENTS

I would like to thank John ‘Jack’ Elliott, Neil Mattson, and Melanie Yelton for their continuing support and mentorship.

Table of Contents

Abstract

Electricity for supplemental lighting is a major cost of year-round crop production, so increasing efficacy of lighting is of primary concern to growers. Market opportunities may support projects with high energy use intensity, which again highlights the need for energy efficient technologies. In greenhouse production, where electric light is supplemented against a broad-spectrum background, only supplemental wavelengths with the highest photosynthetic efficiency justify the cost of electric lighting, and plant growth is largely correlated to the overall quantity of light received. Any reduction in the quantity of supplemental light, while maintaining minimum instantaneous and integrated lighting targets, will coincide with reduced electricity costs for growers. Modulating the intensity of supplemental light can be accomplished at high temporal resolution using computational control algorithms with intensity modulated LED fixtures. The current research compares the accuracy of two lighting control strategies, intensity modulation and binary off/on. Intensity modulation varies supplemental light intensity, balancing supplemental against ambient light to meet an instantaneous light intensity threshold, while binary control switches lights on at full power. Each strategy is combined with the Lighting & Shade System Implementation (LASSI) control algorithm, developed at Cornell University (Albright et al., 2000) which was originally developed for binary control of High Intensity Discharge lights. Binary control is expected to accurately provide a target DLI, with some overshoot on cloudy days that become sunny. Modulation control is expected to precisely meet a minimum threshold DLI while minimizing overshoot of supplemental lighting. Networked microcontrollers were set up to control LumiGrow 650e LED fixtures which were deployed in a greenhouse for 34 days during winter to compare binary vs. intensity modulation control. Overall, this project lays the groundwork for incorporation of additional software into dynamic greenhouse lighting systems.

Background & Motivation

In 1987, the seminal paper "Our Common Future" defined sustainable development as meeting the needs of current populations without compromising the ability of future generations to meet their own needs (Bruntland, 1987). Planetary boundary theory extends the original definition of sustainable development by defining a safe operating range for the parameters that define the biosphere (Steffen, 2015). The planetary boundary framework quantifies the capacity for ecosystems to withstand disturbances within nine categories, including those affected by agriculture (climate change, freshwater use, land system change, etc.). Globally, crop production accounts for approximately 10% of anthropogenic energy consumption, 30% of greenhouse gas emissions, 69 % of water consumption, and the use of 36% of arable land (FAO, 2011; Frenken & Gillet, 2012; Bruinsma, 2017). With the world population predicted to reach 10 billion by 2050, we must produce more food using fewer resources (DESA, Population Division, 2019). Increasing demand coupled with significant resource constraints calls for increasing efficiency in food production. Controlled environment agriculture (CEA) (growing in greenhouses, growth chambers, or indoor farms) affords a superior level of integrated environmental controls that can increase the efficiency of many cropping systems (Albright & Langhans, 1996) with a focus on high nutrient density foods such as vegetables and small fruits. However, some market opportunities, such as demand for fresh, local vegetables in mid-winter, support projects with high resource use intensity, and in some cases, local CEA production can cost more energy than remote field production (Albright & de Villiers, 2008). Extremely energy inefficient indoor crop production (Harbick and Albright, 2016) can be profitable despite the massive cost of growing plants under sole-source lighting (exclusively electric light). Just as appropriate combinations of energy efficient technologies can incrementally reduce energy requirements for commercial and residential buildings, appropriate combinations of energy efficient technologies can incrementally reduce energy requirements for controlled environment agriculture (Albright & de Villiers, 2008; Levine et al., 2007). However, rebound effects complicate the direct benefits of simply increasing energy efficiency. The environmental rebound effect, also

known as the Jevon's paradox (Jevons, 1865), is defined as a reduction in energy savings from technologies that increase energy efficiency, due to increased consumption in response to reduced prices for energy services (Vivanco et al., 2016). Unbounded technological development would seemingly create a classic tragedy of the commons scenario, with savings from increasing energy efficiency unable to outpace increasing consumption (Hardin, 1968). In this view, technological innovations that increase the energy efficiency of food production must be coupled with informed policy that considers agricultural resource dynamics in broader social and economic contexts, emphasizing the application of appropriate technologies to grow more food with less resources (Ostrom, 2002; Orr, 1992). Given that electricity for lighting represents a major input to food production in controlled environments, this paper examines the rationale and investigates one possible solution for increasing energy efficiency of lighting systems in controlled environment agriculture.

The laws of thermodynamics govern ecosystems, whereas profits govern economies. The advertised ecological benefits of indoor agriculture are sometimes misaligned with an accurate accounting of food-system energy use (Benke & Tomkins, 2017). For example, a common misperception is that local indoor farming is more sustainable than remote field production because it eliminates transportation emissions. Although much produce travels over 3000 miles from farm to table, there is undue focus on food miles (Albright & de Villiers, 2008). In fact, local production out-of-season requires significant energy inputs for heating and lighting. For indoor farms, electricity for lighting can account for 40% of the total energy cost of production (Harbick & Albright, 2016). Additionally, plant transpiration presents a significant latent heat load indoors, which requires significant energy for ventilation, dehumidification and re-heating of air (Harbick & Albright, 2016). Field agriculture in remote locations with temperate climate and high solar resource minimizes the cost of this basic agricultural input. Beyond production, oceanic freight and rail transport are extremely efficient. Comprehensive life cycle analysis reveals that the production phase accounts for 83% of agricultural emissions, while the transportation phase

accounts for only 11% (Weber & Matthews, 2008). For comparison, relatively small reductions in average household red meat consumption would reduce emissions more than reductions from localized food production and reduced food miles (Weber & Matthews, 2008). Furthermore, it is unlikely that transportation emissions will ever be completely negligible. If local production is necessary, then the appropriate use of technology and management must fit the given context to justify the allocation of energy resources to CEA production.

Another common misperception is that vertical farming has higher yield per area because of extremely high planting densities in multilayer cropping systems (Ziedler, 2013). A false equivalency is drawn between the footprint of a vertical farm and the same area of field production. A more accurate basis for comparison of yield between systems is canopy area. Assuming equal yields from a single layer, a vertical farm consumes approximately twice as much energy as a greenhouse (Harbick, 2016). The operational energy requirements for both climate control and lighting scale linearly with the number of plants, so adding additional layers only increases the energy requirement. Despite seemingly higher yields per area, vertical farming dramatically increases the energy cost per weight of edible biomass (MJ/kg). Furthermore, the fixed cost and embodied energy of a vertical farm is an order of magnitude higher per area of canopy than for greenhouses (Ziedler, 2013; Albright & de Villiers, 2008). It is simply cheaper to build and operate a larger greenhouse than to increase the number of layers in a vertical farm.

Another common misperception is that vertical farms are orders of magnitude more efficient in water and fertilizer consumption than field production (Benke & Tomkins, 2017). Agriculture accounts for 70% of water consumption in the U.S.; consequently, field agriculture is incorrectly perceived as inefficient with respect to water and fertilizer. The role of evaporation and runoff are exaggerated. Depending on the context (crop type, climate, soil type, etc.), sub-surface drip irrigation achieves between 20-40% reduction in the crop water

requirement compared to surface irrigation (Ayars et al., 2015). Additionally, sub-surface drip irrigation is extremely precise and is typically integrated with automated controls which enable cultivation practices that boost yield like pre-harvest water stress. With field grown processing tomatoes, pre-harvest water stress achieved 200% increase in water use efficiency due to increased yield (Ayars et al., 2015). Biogeochemical flow is defined in an absolute sense within planetary boundary theory, with the major contributor to the eutrophication of aquatic ecosystems being excess N and P from fertilizer application (Steffen et al., 2015). The use of efficient fertigation systems can help maintain the boundary for biogeochemical flows at the global level. Indoor farms often claim higher water efficiency because heat and water are removed through dehumidification and then recirculated to the root zone (Graamans et al., 2018). However, the latent heat of condensation of water (2.26 MJ/kg) makes this process extremely energy expensive (Datt, 2011). After condensation and cooling, the air must be reheated to maintain optimal temperatures in the cultivation space, which takes even more energy (Harbick, 2016). Savings in water and fertilizer efficiency are offset through high energy costs.

Another common misperception is that the use of renewable electricity justifies energy intensive indoor crop production (Benke & Tomkins, 2017). Indirect use of solar energy via renewables to power electric lights is possible; however, energy conversion losses limit the fundamental efficacy of electric lighting powered by renewable electricity (Bejan et al., 2012). Even the most efficient solid-state lighting devices operate at only 40 % efficiency (Tsao et al., 2015). Both indoors and in the field, a mature plant canopy can harness light through photosynthesis with only 5% efficiency (Bugbee & Salisbury, 1988). Efficiency losses at each stage of the entire production system combine to fundamentally limit the efficiency of horticultural lighting powered by renewable electricity. Sole-source lighting may not be the best use of renewable energy when direct use of solar energy through photosynthesis is available at little cost. Latency also remains a significant challenge with renewable production of electricity (Fridley, 2010). Without advances in energy storage technology, continuous operation

of indoor farms will depend on electricity sourced from fossil fuels. Depending on the renewable energy source, other idiosyncratic inefficiencies are apparent. In the case of photovoltaic electricity, production of energy dense crops is a much better use of arable land because plants have evolved to directly metabolize readily available solar energy. The area needed to produce the requisite photovoltaic electricity for sole-source lighting far exceeds the area required to simply grow the crop in the field (Fridley, 2010). Arable land is a fundamental resource boundary defined within planetary boundary theory, and once converted to industrial use, arable land is likely despoiled and no longer suitable for field production (Steffen, 2015). Reducing dependence of food systems on fossil fuels via renewable energy is imperative; conversely, the use of renewable energy does not justify profligate energy consumption when more efficient production technologies are available, i.e. field or greenhouse production that takes advantage of photosynthetically available radiation from the sun.

At mid-latitudes in North America, the summer sun delivers a massive quantity of light energy to the earth's surface. The cost to provide the same quantity of electric light would be approximately $700,000 per hectare (Pattison et al., 2018). Plants grown without sunlight require electric light, which can represent an unreasonable cost burden given that sunlight is freely available in greenhouses and in the field at appropriate latitudes (Harbick & Albright, 2016). Yet certain high value crops can support the cost of growing food with electric light. The suitability of a crop for controlled environment production can be evaluated by calculating its photon cost, or the electricity cost for lighting required to produce a kilogram of yield ($/kg) (Pattison et al., 2018). A crop's photon cost is dependent on a its production efficacy, which is the product of canopy photon capture, quantum yield (photosynthetic efficiency), plant carbon use efficiency, harvest index, and carbohydrate content (Pattison et al., 2018). For example, the photon cost of wheat is approximately four orders of magnitude higher than its fresh market price, whereas the photon cost of lettuce is approximately one percent of its fresh market price (Pattison et al., 2018). The price of wheat cannot support the energy cost of protected cultivation,

whereas the high market price of horticultural specialty crops like lettuce justifies the benefits of controlled environment agriculture (superior quality, proximity to market, year-round production etc.). In certain scenarios and for certain crops, electric lighting is not the appropriate technology, and the costs of electric lighting must be carefully evaluated when assessing the suitability of a crop for controlled environment agriculture.

Controlled environment agriculture includes many different cropping systems that range in production efficiency. While tomatoes produced in a Dutch style greenhouse contain more embodied energy by weight than chicken (35 and 66 MJ/kg, respectively), high tunnel production of tomatoes achieved 2.8MJ/kg (Carlsson-Kanyama, et al., 2003; de Villiers et al., 2011). Large scale field production in locations with a high solar resource achieves extremely low emissions per fresh weight, and only when appropriate energy efficient technologies are combined does local greenhouse production achieve comparably low emissions (Albright & de Villiers, 2008). Local greenhouse production of lettuce in the Northeast can achieve the lowest system energy costs when CO_2 enrichment is paired with high efficiency luminaires, automated lighting and HVAC control, and best crop management practices (Albright & de Villiers, 2008). Co-generation of heat and power and evaporative cooling are other examples of technologies that can enable energy cost parity of local greenhouse production with remote field production. However, the efficiency of CEA production is contingent on certain key assumptions, namely the use of sunlight and the appropriate use of energy efficient technologies given the context. Evaporative cooling technology is cheap and energy efficient but only effective in arid climates and not humid ones and relies on ample water availability. Furthermore, common sense management of cropping systems works in growers' interest to minimize costs. For instance, greenhouse production of horticultural specialty crops with a high light requirement should only occur when the solar resource can provide the majority of input light energy, i.e. during the summer and shoulder seasons. Growers can schedule their production seasonally to reduce their costs. If year-round production is necessary, then selection of crops with lower light integral requirements is appropriate for

the winter months. Agricultural best practices and combinations of energy efficient technologies can incrementally reduce energy requirements for local controlled environment agriculture.

In greenhouse production, where electric light is supplemented against a broad-spectrum background, only supplemental wavelengths with the highest photosynthetic efficiency justify the cost of electric lighting, and plant growth is largely correlated to the overall quantity of light received. Sunlight varies with season, latitude, altitude, and local conditions (Ciolkosz, 2008). Effective management of supplemental lighting must balance the quantity, quality, and timing of electric lighting against variation of light conditions in the ambient environment. A primary goal of any supplemental lighting control is to deliver just enough energy to support optimum growth, while wasting no electricity. Intensity of photosynthetically active radiation (PAR) is expressed in quantum units, or photosynthetic photon flux density (PPFD), which is the number of photons per square meter per second ($\mu mol \cdot m^{-2} \cdot s^{-1}$). The daily light integral (DLI) ($mol \cdot m^{-2} \cdot s^{-1}$) is the sum of PPFD occurring in a day and is commonly used to express crop specific light requirements. Any reduction in the quantity of supplemental light applied, while maintaining minimum instantaneous and integrated lighting targets, will coincide with reduced electricity costs for growers. Modulating the intensity of supplemental light can be accomplished at high temporal resolution using computational control mechanisms and dimmable LED fixtures.

Supplemental Lighting Efficiency

By 1962, advances in semiconductor research led to the development of the first red LED, with orange, yellow, and green following by 1972 (Holonyak & Bevacqua, 1962; Craford et al. 1972; Groves et al., 1971). It was not until 1994, that a high efficiency blue LED was developed following advances in metal-organic vapor phase epitaxy, p-type Mg doping of GaN enabled via thermal annealing, and reduction of semiconductor defect

density via introduction of InN to GaN synthesis (Pust et al. 2015). Blue LEDs enabled efficiency gains across the visible spectrum through phosphor conversion, where a luminescent phosphor coats a shortwave emitting semiconductor, downregulating its radiation to longer wavelengths. These critical advances in semiconductor development were preconditions to the prevalence of solid-state lighting, which is an order of magnitude more efficient than incandescent and twice as efficient as plasma discharge (Tsao et al., 2015). The luminous efficacy (lm/w) of solid-state lighting has steadily progressed over the past two decades. The luminous efficacy of white LEDs has reached approximately 72% of their theoretical maximum, with further increases limited by fundamental material properties of their constituent elements, losses due to reflection and other electronic components (power supply, drivers, etc.) (Haitz & Tsao, 2011) . Predictions vary on the commercial availability of the remaining 28% potential luminous efficacy. If the cost of LED components fall below the cost of the fixture, then further efficacy gains may be prohibitively expensive in the face of only marginal gains. On the other hand, a low fixture cost would ensure that operational cost would dominate a fixture's lifetime cost, incentivizing further gains in luminous efficacy. Unlike plasma discharge lighting, solid-state devices integrate well with computational controls, and their power can be modulated quickly without damage to the device. This ease of integration brings with it an entirely new feature set, including the digital tuning of the timing, spectral quality, and spatial density of LED light output. Indeed, smart control of horticultural lighting offers further production efficiency gains by affecting the timing, quality and quantity of electric lighting.

Of primary concern to growers are the costs associated with supplemental lighting, which are largely determined by two factors: the lighting fixture efficiency and the plant canopy photon capture (Nelson & Bugbee, 2014). Plant canopy photon capture is defined as the fraction of photons transferred to the plant leaves (Nelson & Bugbee, 2014). Lighting fixture efficiency is a measure of light output per energy input, which can be expressed as micromoles of photosynthetically available radiation (PAR) per joule (Nelson & Bugbee, 2014; Both et al.,

2017). While LEDs are generally perceived as more efficient than traditional luminaires, the most efficient high-pressure sodium (HPS) lamps can be as efficient as LEDs (Nelson & Bugbee, 2014). For both LED and HPS, direct use of electricity is the largest contributor to a fixture's lifecycle carbon emissions (Zhang et al., 2017). Many claims are made about the effect of light spectrum on photosynthesis, morphogenesis, and net assimilation; however, light quality has a much smaller effect on plant growth than light quantity (Nelson & Bugbee, 2014). The quantity of light received by a plant is closely tied to canopy photon capture and therefore leaf area index; delivery of supplemental lighting should be precisely synched with plant morphogenesis to maximize its effect on yields (Marcelis et al., 2005). Light intensity is positively correlated with plant growth; however, the relationship is only linear at low to mid range intensity, with declining incremental growth benefit at higher light intensities. The magnitude of the effect diminishes with increasing light intensity and is subject to interactions with leaf temperature and CO_2 concentration (Marcelis et al., 2005). Light emitting diodes emit light at higher intensities than expected from corresponding levels of thermal radiation as predicted by Planck's law, and so are perceived as 'running cool' (Bula et al., 1994). However, all light fixtures generate heat proportionally to their intensity and efficiency at converting electrons into photons, and high intensity LEDs generate significant amounts of heat. Yet in greenhouses, LEDs slightly reduce leaf temperature as compared to HPS (Nelson & Bugbee, 2015). This is because solid-state chips emit a larger fraction of heat away from plant canopy through convection, whereas HPS fixtures emit heat towards plant canopy as long wave radiation. Adequate temperature is necessary for plant growth and development, so increased leaf temperature from radiant energy under HPS lighting may promote more growth than under LEDs (Hernandez & Kubota, 2015). In greenhouse environments, the benefits of excess heat from supplemental lighting are largely context dependent. High-value fruit with low transpiration rates may benefit from reduced temperature, while crops under greenhouse cover in cold climates may benefit from increased leaf temperature (Nelson & Bugbee, 2015). In large, single-layer greenhouses, canopy area is maximized such that electrically efficient HPS top lighting can achieve very low overall system costs

(Gunnlaugsson & Adalsteinsson, 2005). By contrast, in small greenhouses, where aisles and edges take up relatively more space, LEDs can be mounted lower, which focuses photons onto the target area more efficiently without increasing the temperature, greatly improving plant canopy photon capture. The cultivation method, including the specifications of the luminaire (operating temperature, optics), greatly affects the canopy photon capture efficiency and the thermal dynamics of the growing environment. In any case, cost benefit analysis of supplemental lighting in cold climates must consider the heating fuel cost offset afforded by excess heat from luminaires (Dueck et al., 2011; Kubota et al., 2016). The lowest lighting system costs are achieved by combining efficient fixtures with improved photosynthetic photon flux density (PPFD) (Nelson & Bugbee, 2014). Simply put, the light must be delivered to the canopy efficiently, so that no light is wasted, and so that the leaves are not scorched by heat from the luminaire.

Spectral Effects on Plant Growth

While canopy photon capture is fundamental to supplemental lighting efficiency, optimization of the spectral distribution of supplemental light can further improve the efficiency of biomass production (Poulet et al., 2014). However, the effects of light on plant growth and development are species specific, and may interact with other parameters of the light environment such as light intensity, photoperiod, etc. For example, spectral distribution of light can be used to regulate plant morphogenesis and flowering in many herbaceous crops, while there was no significant difference in phytochemical development of Boston lettuce between different light sources (HPS, LED, and sunlight) (Runkle & Heins, 2005; Martineau et al., 2012). Photosynthetically active radiation (PAR) describes the wavelengths between 400 and 700 nanometers, however quantum yield varies across this range (McCree, 1972). Quantum yield is a wavelength specific measure of photosynthetic efficiency, expressed as moles of carbon fixed per mol of photons absorbed (McCree, 1972). Quantum yield peaks in the red and blue

spectra where carbon fixation is most efficient, i.e. the least number of photons are required per mol of carbon fixed. (McCree, 1972). In greenhouses, where electric light supplements a high intensity background, the wavelengths with the highest quantum yield are the most economical because solar radiation provides wavelengths that are less electrically efficient (Bugbee, 2016). However, with sole source lighting, the exclusive use of the optimal spectra may preclude biotic effects of other bands of light (ultraviolet, green, far-red, near-infrared). The absolute spectral environment interacts with changes within a narrow bandwidth with respect to morphogenesis due to plant photoreceptors (Brown et al., 1995). The integrated product of spectroradiometric data from a given light source and the relative quantum yield of photosynthesis provides the yield photon flux curve (YPF), a more precise measure of actinic light than PAR (Sager, 1988). The YPF curve attenuates blue and amplifies red wavelengths, reflecting photosynthetic efficiency across the actinic spectrum. Measures of phytochrome photo-equilibrium describe expected morphogenic and photosynthetic responses of many plant species relative to specific spectral environments (Sager, 1988). Where the spectral environment affects morphogenesis, any attenuation in leaf area will negatively impact growth, because carbon assimilation is positively correlated with radiation capture efficiency, which is dependent on leaf area (Bugbee, 2016). Overall radiation capture, or the proportion of light absorbed, has the highest impact on biomass gain. Comparing the McCree curve to the wavelength dependent electrical conversion efficiency of LEDs reveals that red light (668 nanometers) is the most efficient, both photosynthetically and electrically (McCree, 1972; Tennessen et al., 1995).

There is no apparent consensus on the effect of green light on growth and quantum yield. Green light is known to penetrate the plant canopy and drive photosynthesis among lower leaves, yet reports have both confirmed and denied the null hypothesis on effects of green light on plant growth and quantum yield. (Kim et al., 2004; Snowden et al., 2016; Terashima et al., 2009). Mutual shading changes the spectral distribution of light reaching lower leaves, resulting in green light enrichment. In the lower canopy, reductions in light intensity induces leaf

senescence more than changes in light quality (Frantz, 2000). Additionally, variance in the quantum efficiency across spectra when measured on single leaves at low PPFD over short intervals, may not generalize to whole plants or plant communities (Snowden et al., 2016).

Deficiency or excess of blue light is detrimental to plant growth and development. Small fractions of blue light are beneficial to plant growth (Massa et al., 2015). Blue light is necessary to prevent shade avoidance response in many but not all species, yet excess blue light reduces hypocotyl length, stem length, petiole length, and leaf area, which then inhibit radiation capture and growth (Cope et al., 2014). The absolute photon flux of blue light, not the blue-red ratio, controls photomorphogenic shade avoidance (Bula et al., 1994). Response to blue-red ratios is species specific and dependent on intensity of the solar background (Hernandez & Kubota, 2015). Combined blue-red sole source lighting induced more plant growth than red light alone (Hernandez & Kubota, 2015). While high-intensity blue light can reduce growth, low-intensity blue against a red background supported as much growth in *Capsicum* as a full spectrum light environment and more growth than monochromatic red light (Brown et al., 1995). Increased blue fraction of supplemental lighting offers morphogenic control in some species, inhibiting stem extension and compactness (Islam et al., 2012). Spectral control of morphogenesis can be more energy efficient than a temperature-drop strategy in producing compact plants because it requires no ventilation (Islam et al., 2012).

Lighting Targets - Daily Light Integral, Intensity, Photoperiod

At northern latitudes, where solar radiation is severely limited during the winter months, supplemental lighting is necessary to support year-round production. The life cycle energy cost (embodied, direct use, transport) of local crops produced out-of-season in the northeast can be much greater than imported field grown crops (Albright &

de Villiers, 2008). Direct use of electricity for supplemental lighting can account for approximately half of the total energy required to produce a crop (Albright & de Villiers, 2008). Precise control of light quantity and quality is necessary to efficiently translate supplemental lighting into increased yields. Daily light integral (DLI), intensity and photoperiod are the basic parameters of the quantity of light received.

Light Intensity & Photoperiod

Light intensity, or photosynthetic photon flux density (PPFD), is the number of photons occurring per unit of space per unit of time (Thimijan & Heins, 1983; Barnes et al., 1993). Equal integral lighting describes the inverse relationship between intensity and photoperiod when DLI is held equal. While plants will respond to average irradiance over a given photoperiod, low-irradiance, long-day supplemental lighting can be more photosynthetically efficient than high irradiance short-day because of the hyperbolic relationship between light intensity and photosynthesis, resulting in lower quantum efficiencies at high intensities (Adams & Langton, 2005). However, variable electricity pricing favors instead high-intensity, short-duration supplemental lighting during the off-peak period. Lights with higher maximum PPFD can deliver more light when electricity is less expensive, anticipating tradeoffs between optimized plant quality and electricity prices (Clausen et al., 2015). On the other hand, high intensity supplemental lighting at long duration can induce vegetative growth instead of generative growth, which, in the case of a fruiting crop, does not support the cost of electricity (Heuvelink et al., 2005). Night break lighting is low-intensity supplemental lighting for 1 to 4 h applied in the middle of the daily dark period associated with a short day; night break lighting has both qualitative and quantitative effects on flowering in different species (Adams & Langton, 2005; Runkle & Heins, 2005).

Daily Light Integral

The daily light integral (DLI) is the sum of the light occurring in a day; DLI is measured in $mol \cdot m^{-2} \cdot d^{-1}$ (Korczynski et al., 2002). Across the contiguous United States, mean DLI ranges from 5 to 60 $mol \cdot m^{-2} \cdot d^{-1}$, varying with latitude and season (Korczynski et al., 2002). At mid-range to lower light intensities, any reduction in the quantity of light will correspond with reductions in yield (Cockshull et al., 1992). Therefore, crop specific DLI targets describe precisely the minimum quantity of light needed to optimize time to harvest without wasted electricity. For example, lettuce requires at least 17 $mol \cdot m^{-2} \cdot d^{-1}$ to harvest within 24 days from transplanting (Both et al., 1994). Lettuce can withstand relatively small deviations from optimal DLI (17 ± 3 $mol \cdot m^{-2} \cdot d^{-1}$) over the course of 3-days without compromising crop growth (Albright et al., 2000). However, tip-burn in lettuce is a physiological condition where high radiation and low transpiration causes calcium deficiency along meristem edges (Frantz et al., 2004). Therefore, precise control of both DLI and light intensity is necessary to prevent physiological disorders, minimize harvest times, and maximize yield.

Latitude and altitude can be used to predict DLI for a given location via a percentile exceedance value calculated from typical meteorological year data (Ciolkosz, 2008). Percentile exceedance is the proportion of days in a year that exceed a given DLI value. A statistical model enables DLI predictions for locations with no empirical data; however, the model makes assumptions about local conditions and the diffuse fraction of light (Ciolkosz, 2008). Local conditions producing higher or lower diffuse fractions of light will significantly impact the quantity of light reaching plant canopy within a greenhouse. Further analysis is needed to further parameterize DLI predictions with respect to local conditions. DLI is the accepted horticultural standard measurement for light requirements of a crop. Predicting DLI is necessary for growers to maintain optimum conditions for their crops while avoiding excess electricity cost burden.

Lighting Control Strategies

LEDs are solid state devices that afford a high degree of digital integration, which makes them well suited for computer controlled, dynamic optimization of supplemental lighting. Dynamic optimization of greenhouse systems can be achieved through the integration of greenhouse, crop, and heuristic models into algorithms that make control decisions dependent on economic, environmental, biological, and energetic factors (Challa et al., 1988). Each model should quantitatively describe a component of the cultivation system, generating control signals that account for dynamic interactions in the growing environment. Features can be incorporated from many domains including hardware specification, real-time weather data, and biological properties of the crop. Control decisions should be made in the order of minutes to account for dynamic interactions between subsystems (Challa et al., 1988). The simplest lighting control strategies include scheduling, thresholding, and power modulation.

Scheduling & Thresholding

During scheduling control, the DLI requirement, photoperiod, and PPFD target are calculated manually and the lights are scheduled to be on accordingly. Scheduling is particularly well suited to crops with a dark requirement, when the lights must be off to promote photoperiodic effects. Scheduling is advantageous due to its simplicity, without the need for specialized equipment. However, often scheduling simply takes place according to a preset time clock which can lead to the wasted addition of electrical light on sunny days or the target DLI not being reached on cloudy days.

With threshold control, the lights are automatically turned on if ambient light levels fall below a specified PPFD target. Controlling greenhouse lighting with a light sensor enables supplemental lighting to react

dynamically to changes in the ambient light environment. If power modulation is not available, intensity is fixed, and the fixture height must compensate for the PPFD target at canopy level. Thresholding control can potentially miss the PPFD target depending on the light output from the fixture. Depending on the temporal resolution of the control program, this imprecision accumulates, resulting in failure to meet the DLI target. When the difference between the PPFD threshold and ambient light levels is greater than PPFD from lights at full power, thresholding control will overshoot the DLI target (Harbick et al., 2016). When thresholding with HPS lamps, a delay must accompany ignition to avoid too many on/off cycles which would result in premature degradation of the fixture. LEDs bear the distinct advantage over HID fixtures in that they can be cycled without producing undue wear on the fixture.

Rapid Cycling or Power Modulation

Rapid cycling, or pulse width modulation (PWM), describes a technique of modulating power to a device such that the device is cycled between its on and off state extremely rapidly (Tennessen et al., 1995). When power to a device is modulated, the proportion of time that the device is powered is called the duty cycle. Low duty cycle results in the appearance of the light dimming, when in fact the light is being cycled rapidly between on and off states (Pinho, 2013). Equal integral lighting delivered in microsecond pulses did not affect photosynthetic efficiency as compared to continuous photon flux, demonstrating that PWM can support photosynthesis (Tennessen et al., 1995). Modulating the power to an LED fixture enables the fixture to meet a PPFD threshold more precisely. The fixture provides only as much supplemental light as is required to meet the target and no more. Compared to scheduling and thresholding, power modulation can significantly increase the efficiency of supplemental lighting depending on PPFD target and solar DLI (van Iersel and Gianino, 2017). Precise control over light intensity enables a range of control strategies that include balancing tradeoffs between quantum

efficiency and electrical efficiency; night-break lighting to affect flowering; and precisely balancing solar and supplemental lighting to meet DLI requirements (see Light Intensity & Photoperiod).

Biofeedback

Biofeedback systems can be used to control supplemental lighting based on photosynthetic efficiency, radiation capture efficiency, net assimilation rate, and electron transport rate. Because electricity is expensive, it is important to deliver supplemental light only when plants can use it efficiently. Biofeedback systems can be used to control light intensity based on photosynthetic efficiency. Van Iersel et al. (2016) used chlorophyll fluorescence to modulate light intensity and meet target electron transport rates. Kjaer et al. developed a lighting control program dependent on daily photosynthesis integral (DPI) ($mmol \cdot CO_2 \cdot m^{-2} \cdot leaf \cdot day^{-1}$). The timing of photosynthesis is well understood: primary photochemistry occurs at picosecond to nanosecond speeds; electron shuttling between photosystems occurs at microseconds to millisecond speeds; carbon metabolism within the chloroplast occurs within seconds, while sucrose metabolism and enzyme activation takes minutes (Tennessen, 1995). Digital control programs should precisely synchronize supplemental lighting with photosynthesis and plant metabolism, yet controlled environments are subject to multiple and dynamic interactions between climatic parameters that may complicate phytochemical models of plant growth. For example, carbon assimilation generally increases with overall light quantity, yet excess radiation also causes photoinhibition and reduces photosynthetic efficiency (Tennessen, 1995). Furthermore, net photosynthesis measured at the leaf level ($mmol \cdot CO_2 \cdot m^{-2} \cdot leaf^{-1} s^{-1}$) may not actually reflect net assimilation rate at the crop level (g of dry mass $m^{-2} \cdot day^{-1}$) (Bugbee, 2016). Further research is needed to appropriately quantify and digitally affect the biological parameters of plant growth.

Economic Models

Economic models can provide lighting control signals based on dynamic cost-benefit analysis, breakeven intensity values, and crop specific efficacy of supplemental lighting. Crop specific efficacy measures the dollar return per mol of PAR delivered (Kubota et al., 2016). A model of crop specific efficacy accepts the following parameters: lamp photon efficiency, electricity cost, canopy photon capture efficiency, and heating fuel cost offset (Kubota et al., 2016). The breakeven intensity for supplemental lighting is the PPFD value for which the cost of delivery equals the return from increased yield (Heuvelink & Challa, 1989). The breakeven light intensity value is primarily dependent on CO_2 concentration and electricity price, but also on product price, leaf area index, and ratio of harvestable product to gross fresh weight (Huevelink & Challa, 1989). Carrier et al. developed algorithmic control of supplemental lighting that notably incorporates cost-benefit analysis into lighting decisions. The program computes anticipated profit or loss from supplemental lighting as a function of marginal dry weight, product price, and the costs of electricity and packaging (Carrier et al., 1994). A source of error arose from the fact that food prices may vary faster than it takes a crop to grow, and that electricity prices may vary faster than the decision step of a lighting control program. The underlying variance of a model's independent variables must be normalized with respect to each other and its dependent variables.

Lighting & Shading System Implementation (LASSI)

The Lighting and Shading System Implementation (LASSI) is a lighting control algorithm that consistently meets DLI targets, without need for machine learning or weather data (Albright et al., 2000). A series of rules makes hourly lighting decisions, based on predicted solar DLI, the potential for supplemental lighting, the season and the time of day. The algorithm defers supplemental lighting as much as possible to the off-peak period, occurring during the night and early morning when electricity rates are cheaper. The algorithm takes effect by

delaying supplemental lighting each day for a number of hours which is determined by the season and the amount of insolation occurring to the current time. Delaying the start of supplemental lighting prevents excess lighting that may occur during months of greater solar irradiation when cloudy morning hours are followed by sunshine. LASSI meets DLI targets more accurately than simple thresholding control, and as a result saves energy while maintaining adequate light for growth (Harbick et al., 2016). On days when insolation increases steadily from the morning through the afternoon (initially cloudy days that become sunny), supplemental lighting can occur early in the day despite the capacity of the sun to meet or exceed the DLI target. In these cases, supplemental lighting is unnecessary and can even lead to a shading requirement. One of the primary benefits of greenhouses compared to sole source lighting environments is the energy cost offset provided by free solar radiation (Harbick & Albright, 2016). Supplemental lighting that triggers a shading requirement diminishes this advantage. Seginer et al. (2005) modified LASSI to meet a three-day target light integral, which compensates for single-day deficits with excess solar radiation from the next two days. The three-day target effectively balanced excess insolation without a shading requirement and therefore reduced the need for supplemental lighting.

The current research compares the efficiency of two lighting control strategies, intensity modulation and binary off/on. Intensity modulation varies supplemental PPFD, balancing supplemental against ambient light to meet an instantaneous PPFD threshold, while binary control switches lights on at full power at much longer time steps (ex., every hour). Each strategy is combined with the Lighting & Shade System Implementation (LASSI), developed at Cornell University (Albright et al., 2000). The LASSI algorithm strives to meet a target daily light integral, making lighting decisions based on expected solar DLI, the potential for supplemental lighting, the season and the time of day. The algorithm takes effect by delaying supplemental lighting each day for a number of hours determined by the season and the insolation occurring to the current time. Previously, LASSI used binary control to account for the long warm-up time and reduction in bulb-life associated with repeated on/off cycles for

High Intensity Discharge (HID) fixtures, such as High Pressure Sodium (HPS) which have been traditionally used in greenhouse lighting. In this project networked microcontrollers running either a binary or intensity modulation control strategy manage LumiGrow 650e LED fixtures. This project lays the groundwork for incorporation of additional software into dynamic greenhouse lighting systems.

Methods

For the current research, quantum light sensors controlled two Lumigrow 650e LED fixtures via Photon microcontrollers (LI190-R, LICOR, Lincoln, NE; Lumigrow, Emeryville, CA; Particle Industries, San Francisco, CA). Instantaneous light intensity ($\mu mol \cdot m^{-2} \cdot s^{-1}$) was measured with a quantum light sensor (LICOR, Lincoln, NE) which was placed below the light fixtures and thus measured combined supplemental and ambient light. The microcontrollers were loaded with an adapted version of the Lighting and Shading System Implementation (LASSI); the same program was instantiated twice with one critical difference controlling the way the lights are powered. Using binary control, the light was always switched on at full power, whereas using modulation control, just enough power was supplied to the light to meet a predetermined intensity threshold. As different lighting conditions triggered the rules of the LASSI algorithm, the program made decisions about the timing and intensity of supplemental light. If the intensity threshold was low, then modulation control compensated by extending the photoperiod to meet the DLI target. If the intensity threshold was too high, then modulation control responded with a shorter photoperiod. Daily light integral (DLI) was tracked from sunrise to sunrise and used as a critical metric of evaluation of each program's performance. Concurrent observations of binary and modulation lighting controls compared the accuracy with which each program met the target daily light integral (DLI).

Experimental Design

Two independent variables varied between treatments, the method of powering the light (binary or modulation), and the light intensity threshold (100 – 500 µmol· m^{-2}·s^{-1}). The sampling distribution was the range of possible daily light integral (DLI) values occurring for Ithaca, New York, USA in a year. Average DLI values in Ithaca during the winter months range from 10-15 mol· m^{-2}·d^{-1} outdoors, however there are frequent days where DLI drops below 10 mol· m^{-2}·d^{-1} (Korczynski et al., 2002) (NREL). In addition, greenhouse transmissivity is often 50-70% so winter DLI can be further limiting. Data was collected during the 34 days between January 7 and February 9, 2018 at a one second resolution. Matched pairs of samples established a t-interval, measuring the difference in accuracy between lighting controls in meeting a target DLI.

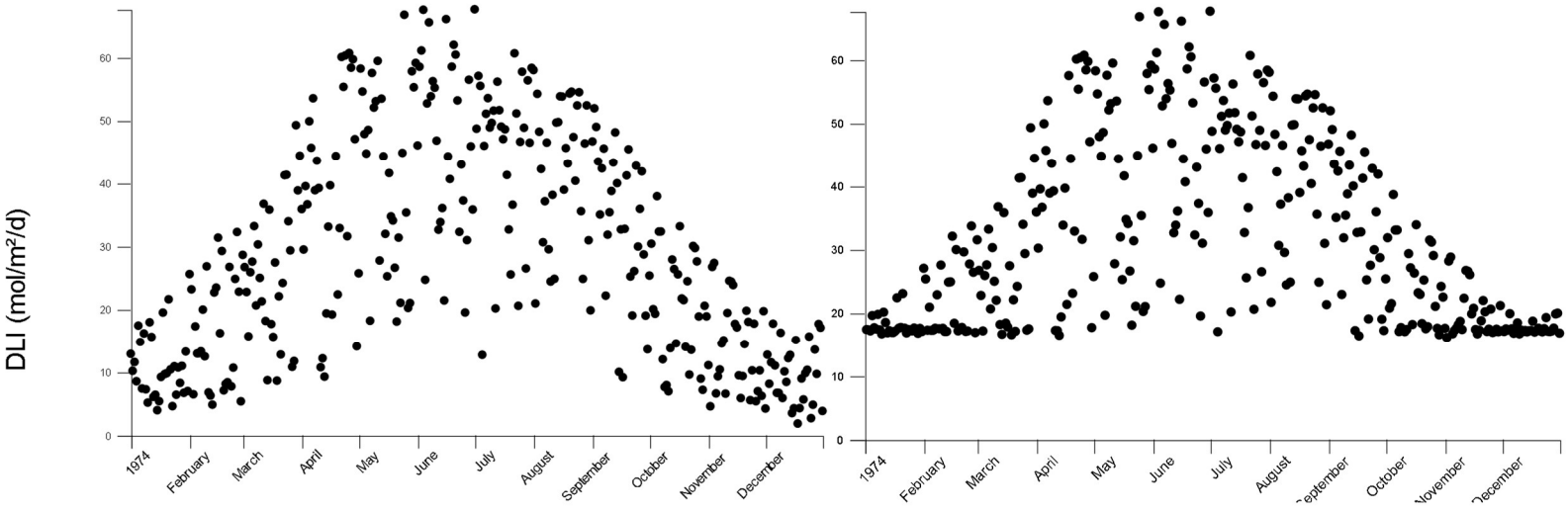


Figure 1

National Solar Resource Database (NSRDB) irradiance dataset for a typical meteorological year in Ithaca, NY. Each black circle represents integrated light for one day of the year. The NSRDB data was used to validate this implementation of the LASSI algorithm. LEFT: Unadjusted data. Note the normal distribution of sunlight throughout the year, as well as high variance, with very cloudy days occurring even in the middle of summer. RIGHT: Data adjusted with LASSI algorithm. Supplemental lighting is simulated for all days where DLI falls below the target DLI of 17 µmol· m^{-2}·s^{-1}. The shading portion of the algorithm was excluded.

The experiment tested five intensity thresholds between 100 – 500 µmol· m^{-2}·s^{-1} in increments of 100 µmol· m^{-2}·s^{-1}. Pseudo-random selection of intensity thresholds occurred daily. Each day, both binary and modulation conditions selected the same intensity threshold via the modulus of the date (1-365) and the number of intensity thresholds (5). The result was the sequential selection of intensity thresholds on a daily basis. Intensity thresholds were not repeated with fewer than 5 days delay. Selection of intensity thresholds in this manner was a pseudo random attempt to disrupt confounding effects of contiguous and interdependent weather patterns. This could also have been achieved via replication over time and space; however, this was not possible given the logistics of the current research. At 42 degrees latitude, both treatments were aligned with the south-facing solar angle, so they would experience similar conditions. Blackout curtain separated the conditions to prevent contamination of the light environment from the LED fixtures.

Quantum Light Sensors

A quantum sensor is a photodiode that produces a linear response in microamps to photosynthetic photon flux (LICOR, Lincoln, NE). A microcontroller is a small, versatile computer that can be used to receive and log data from sensors, transmit data between devices, and control devices with custom software. To adapt the quantum sensor to the microcontroller, it was necessary to convert the microamp output of the sensor's photodiode to the appropriate input voltage range of the microcontroller with respect to the maximum expected PPFD. This can be achieved using an operational amplifier (OP90; Analog Devices, Norwood, MA). According to Ohm's law, the resistance between an operational amplifier's input and output determines the output voltage range (Mancini, 2003). A calibration constant accompanies each quantum sensor from the manufacturer, which compensates for normal variance in the sensor's optical instruments. The calibration constant describes a sensor's response to a known input in microamps per 1000 µmol (Licor, Lincoln, NE). The sensor's expected response to the maximum

possible PPFD can be calculated from its calibration constant (1). The Photon microcontroller accepts input within the standard voltage range of 0-3.3V (Particle Industries, San Francisco, CA). Using Ohm's law, we calculated a feedback resistor value that scales the sensor's maximum expected response in microamps to the maximum input voltage for the microcontroller at 3.3V (2) (Phillips & Bond, 1999).

$$(1)\ \frac{7.28\ \mu A}{1000\ \mu mols} \times 2500\ \mu mols = 18.2\ \mu A$$

$$(2)\ \frac{3.3\ volts}{18.2\ \mu A} \times \frac{1000000\ \mu A}{1A} \times \frac{1\ \Omega}{1000\ k\Omega} = 181.3\ k\Omega$$

The calibration constant for the quantum sensor used in this experiment was 7.28 µamp per 1000 µmols. The quantum sensor's expected response for a maximum expected PPFD of 2500 µmol· m^{-2}·s^{-1} was 14.56 microamps. A resistor value of approximately 180 kΩ forced the sensor output into the microcontroller's input voltage range.

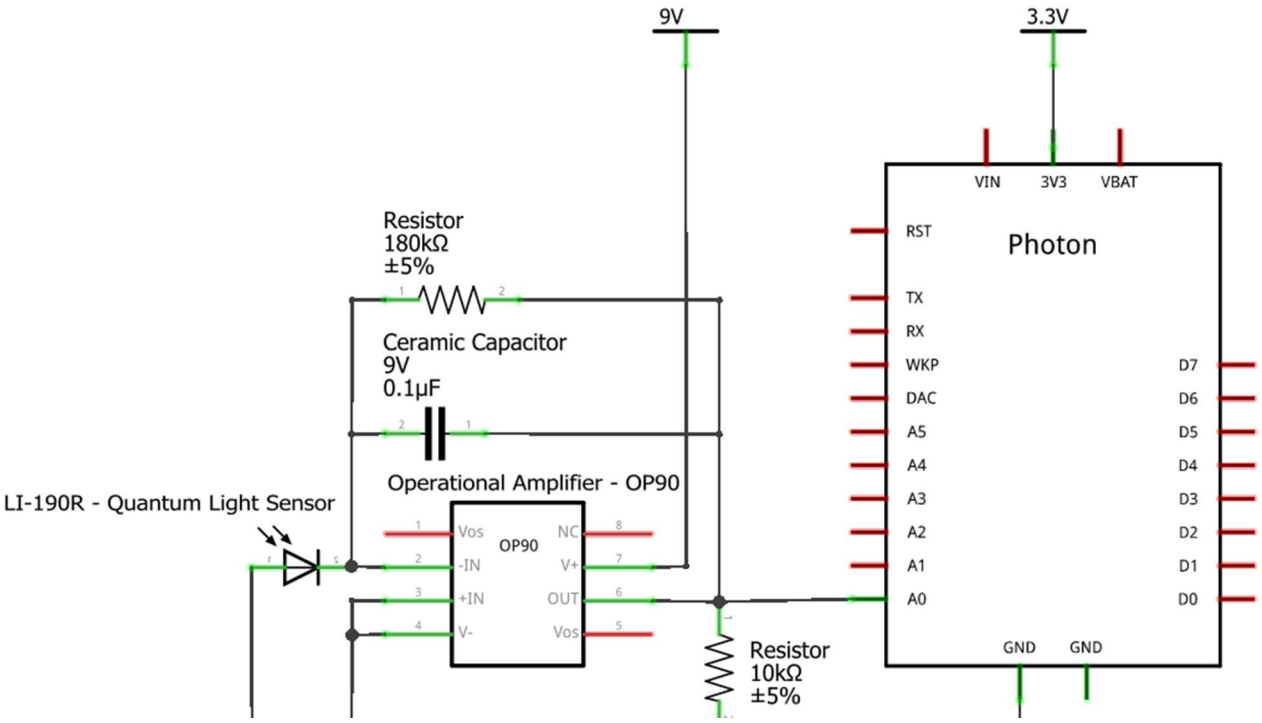


Figure 2

Circuit adapted from Phillips & Bond, 1999. The circuit adapts a LICOR LI-190R quantum light sensor with bear leads to a Particle Photon microcontroller. The resistor is sized to the calibration constant specified by the manufacturer. An operational amplifier maps the milliamp signal coming from the quantum sensor into a voltage range acceptable to the microcontroller.

The microcontroller had 12-bit analog read resolution for input values ranging from 0 - 3.3V. A low pass filter reduced noise inherent to the signal, and the input voltage range was mapped to the output range of 0 – 2500 µmols. Both quantum sensors were calibrated against a constant intensity and spectrum, adjusting the output to match a third, well-calibrated quantum sensor.

Datalogger & Network Communication Protocols

The Lighting and Shading System Implementation was set to run locally on each microcontroller. A small amount of data could be logged to local memory; however, larger quantities of data had to be logged with expanded memory or a networked database. The Photon microcontroller firmware natively supports network communication protocols including HTTP and WebSockets.

In conjunction with a webserver, the microcontroller used HTTP to stream and log global horizontal irradiance data from the National Solar Resource Database (NSRDB) (NREL, 2018). A conversion factor of 4.57 $\mu mol \cdot m^{-2} \cdot s^{-1}$ per W m^{-2} can be used to convert PAR between units of irradiance and photon flux density (Thimijan & Heins, 1983). Note that this is only accurate for wavelengths between 400-700nm. As LASSI made supplemental lighting decisions based on the amount of current insolation and user-defined lighting targets, each rule of the algorithm was logged when it was triggered. A histogram of rules triggered allowed the validation of this implementation of LASSI, revealing when and why lighting decisions were made. This permitted fine-tuned adjustments to the algorithm's parameters that reduced variance around the DLI target. The dataset had a half-hour timestep.

Finally, the microcontroller used WebSockets to stream and log experimental data from the greenhouse environment to a networked database. WebSockets offer a low-latency, real-time communication interface, capable of efficiently transmitting data from firmware to a client via a server. Data was logged at a one second resolution and visualized on a website (Duffy, 2018).

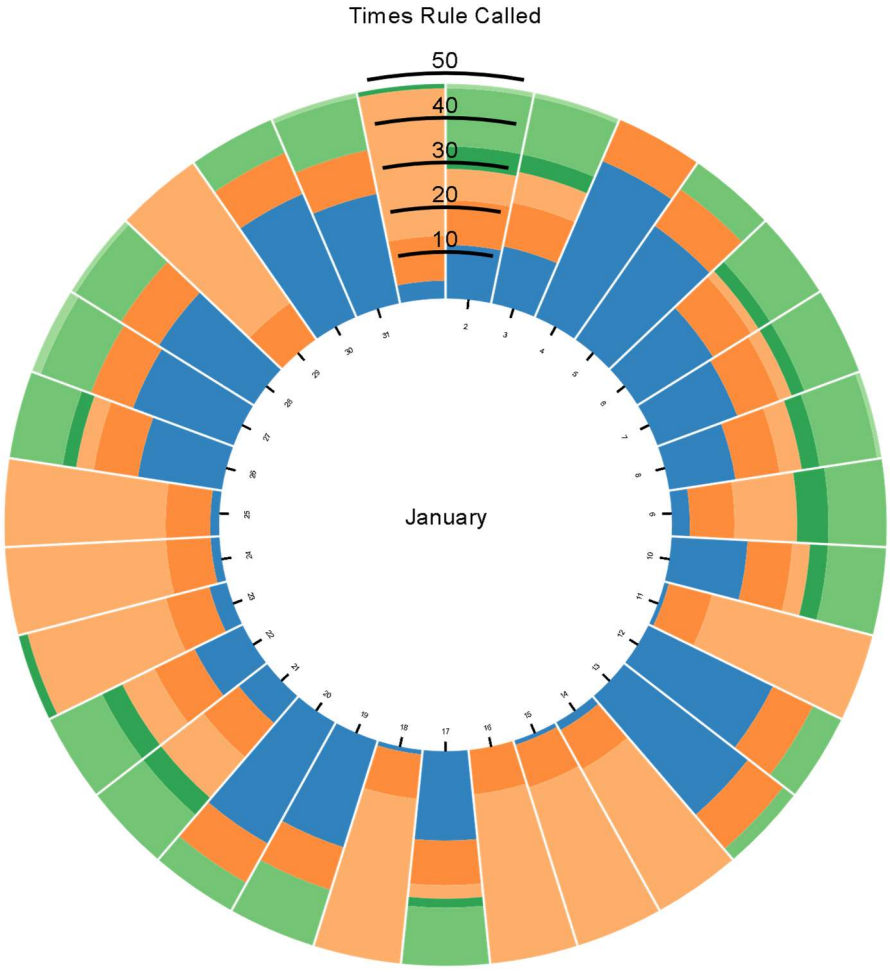


Figure 3

A histogram of the rules of the LASSI algorithm triggered by ambient conditions for January of a typical meteorological year. Each color corresponds to a different rule of the LASSI algorithm. Note the blue area represents the number of hours the lights were on for January. In June, the same figure would have little to no blue area, because sunlight was adequate to meet DLI target. See the color coded rules listed below. Both these figures are represented here as they appeared on the website for the project.

Algorithm Rules

Albright, L. D., Both, A. J., & Chiu, A. J. (2000). Controlling greenhouse light to a consistent daily integral. *Transactions of the ASAE, 43* (2), 421.

Rule 0	Lights on.
Rule 1	If time clock control is included and the current hour is during the period when lamps should be off, control is activated and the lamps are turned off.
Rule 2a	For months of greatest solar irradiation, keep lamps off between sunrise and 12 hours after sunrise. However, if the daily accumulated PPFD is not equal to at least one-quarter of the daily target by solar noon, permit lights to remain on.
Rule 2b	For late summer (when days are still sunny, but solar intensity has lessened), keep lamps off between sunrise and 9 hours after sunrise. However, if the daily accumulated PPFD is not equal to at least one-quarter of the daily target by solar noon, permit lights to remain on.
Rule 2c	For spring and autumn months, keep lamps off between sunrise and 7 hours after sunrise.
Rule 2d	For the rest of the months of the year, keep lamps off between sunrise and 2 hours after sunrise.
Rule 3	If solar PPFD accumulated to this hour meets or exceeds the accumulation target (eq. 6) for the hour, turn the lights off.
Rule 4	If the hour is during the time of year with more sunlight and between sunrise and sunset, and the PPFD left to be accumulated can be achieved by delaying supplemental lighting until the next hour even if solar PPFD drops suddenly to insignificance, and the PPFD deficit to this point could be made up by a scaled portion of the off-peak PPFD potential, turn off the lights.
Rule 5	Turn off the lights if the hour is between sunrise and sunset and the PPFD left to be accumulated could be accumulated by turning on the lights at the next hour even if the solar PPFD drops immediately to insignificance and remains there for the rest of the day.
Rule 6	If the hour is at sunset or between sunset and an hour before the start of off-peak electric rates and the accumulated PPFD deficiency to this hour could be achieved during off-peak hours alone, turn off the lights.
Rule 7	If the hour is before off-peak electric rates start, but any remaining PPFD to be added by supplemental lighting will be achieved before the off-peak period ends, turn off the lights.
Rule 8	Be sure lights are not turned off if the hour is during the dark part of the year and there remains more integrated PPFD to be added than can be met by the lamps, alone, operating from the next hour until the following sunrise.

Web Interface & Data Visualization

Data visualization for the project was developed as a web-app using Node.js, a JavaScript runtime environment. The server is structured using RESTFUL API design and performs data preprocessing before serving assets to the client (Narumoto, 2018). Node.js has extensive support for the incorporation of open-source packages into a web application. Below is a list of the essential packages used in this project:

- Bluebird – third-party promise library for synchronous callback execution - http://bluebirdjs.com/docs/getting-started.html
- Bootstrap – responsive frontend webdesign - https://www.npmjs.com/package/bootstrap
- Browserify – javascript dependency bundler - http://browserify.org/index.html
- Chai – javascript assertion library - http://www.chaijs.com/
- D3.js – extensive javascript data visualization library - https://d3js.org/
- Express – web application framework and simple asset server - https://expressjs.com/
- Express-ws – websocket integration for express - https://www.npmjs.com/package/express-ws
- Grunt – javascript task runner, workflow automation - https://gruntjs.com
- Jquery – clientside javascript integration - https://jquery.com/
- Lodash – javascript array manipulation - https://lodash.com/
- Mongodb – document oriented database - https://www.mongodb.com/
- Mongoose – object modeling for mongodb - http://mongoosejs.com/
- Particle-api-js – clientside firmware integration - https://docs.particle.io/reference/javascript/
- Particle-cli –firmware command line interface - https://docs.particle.io/reference/cli/
- Suncalc – astronomy toolkit for calculating sunrise/sunset - https://www.npmjs.com/package/suncalc
- Ws – websockets for nodejs - https://www.npmjs.com/package/ws

Data Analysis and Interpretation

. A sample of 14 days were deemed valid for the establishment of a t-interval (see results), which estimates the mean difference in accuracy between binary and modulation controls in meeting a DLI target ($17\ mol \cdot m^{-2} \cdot d^{-1}$).

Prerequisite to a t-interval, the sample must be random, normal, and independent. It is reasonable to assume a degree of independence between non-consecutive days included in the sample. The intensity threshold treatment was applied pseudo randomly, such that intensity threshold treatments were repeated every five days. This was intended to break dependence between treatments on contiguous weather patterns. The experiment was carried out once during the winter months in the northern hemisphere, which limits the degree to which the results can be extrapolated to other times of year. However, the need for supplemental lighting in growing environments is greater during the winter months, so the effect of increased accuracy of supplemental lighting controls may be more impactful during this period.

The relatively small sample size (14) requires examination of the distribution of the differences in accuracy between binary and modulation controls in meeting the DLI target. A Jarque-Bera test confirms the distribution of the sample is likely normal ($p=0.003$). The distribution of differences is symmetric with no outliers.

Results

The primary motivation of the current research is to test the hypothesis that modulation control of supplemental lighting was more accurate in meeting a DLI target than binary control, and that the increase in accuracy can be expressed in quantum units (mol· m^{-2}·d^{-1}) as well as an electricity cost to the grower.

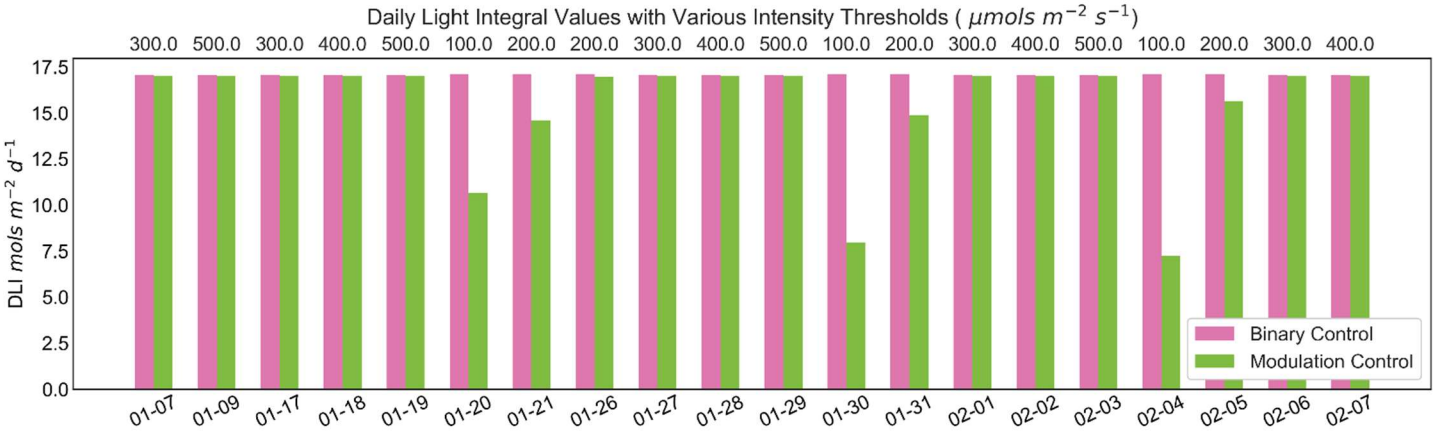


Figure 4

DLI values for both modulation control (green) and binary control (pink) for 20 days in January and February of 2018. The measured DLI included both sunlight and supplemental light in a greenhouse. Note that the days for which modulation control was deficient in meeting the DLI target were days where the PPFD target was set so low that the modulation control could not possible meet the DLI target within 24 hours. This demonstrates the capacity of modulation control to respond automatically using the LASSI rules as well as to user defined settings.

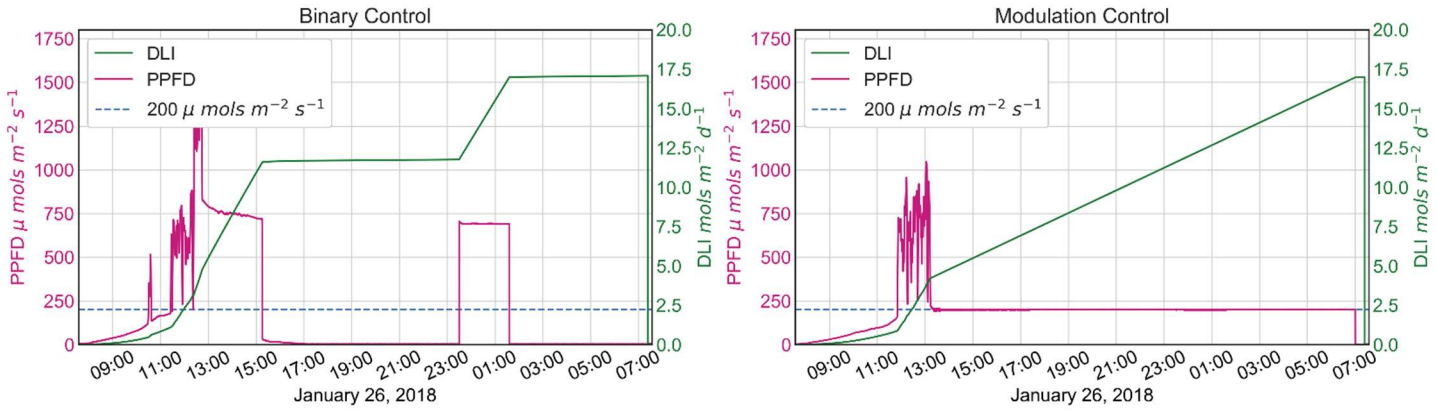
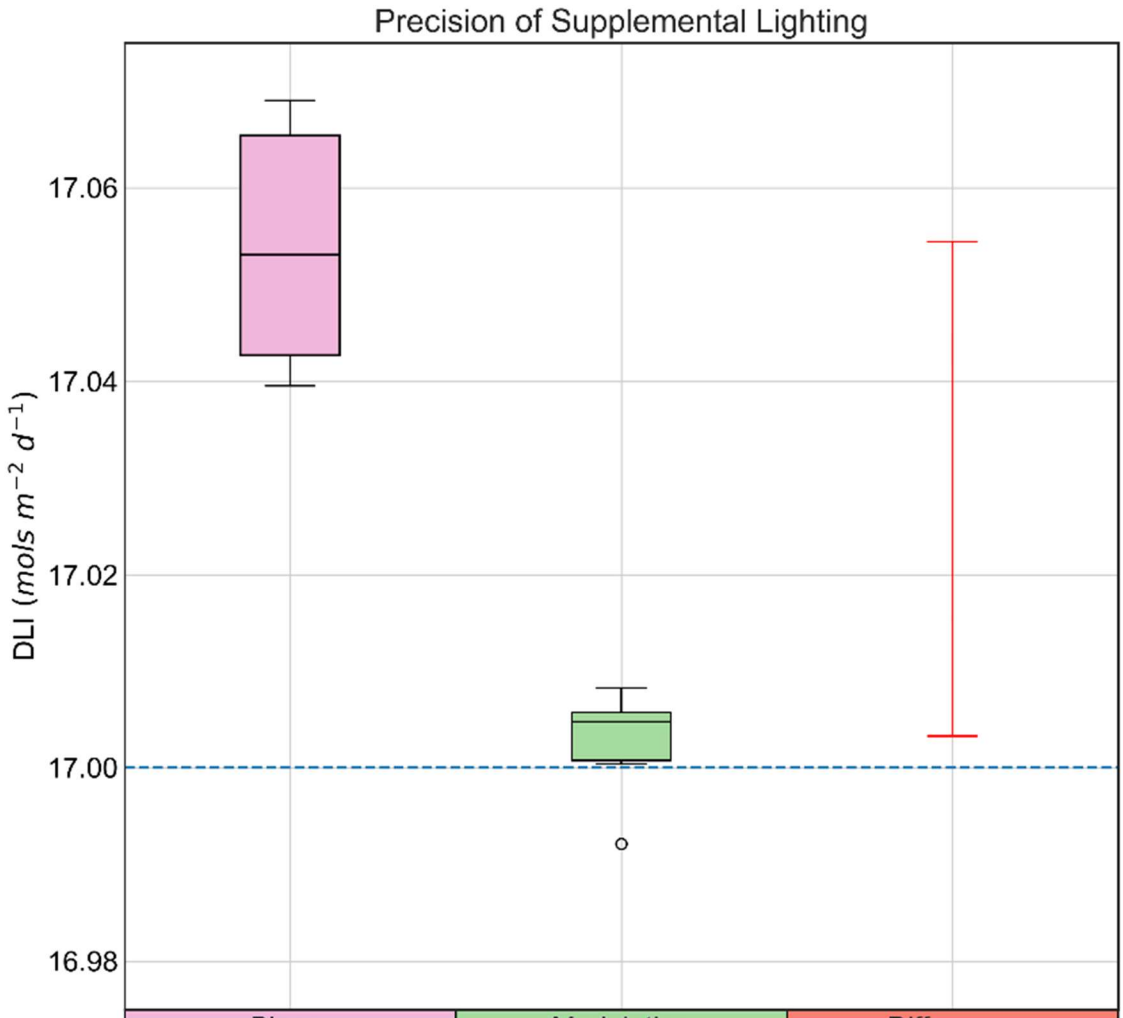


Figure 5

Light intensity was recorded on a one second interval for 34 days. A single day is presented above for both modulation control (right) and binary control (left). Light intensity (pink solid line) and integrated light (green solid line) are shown. The intensity threshold for this day was set to 200 µmol· m^{-2}·s^{-1} (blue dashed line). Binary control responded with short duration high intensity lighting, while the modulation control responded with low intensity long duration lighting. Both control schema met the DLI target with precision and accuracy.

Attributes of the Sample

The final sample is composed of the sequence of 34 days between January 7 and February 9, 2018, for which 20 days were excluded, leaving 14 days in sample. Days were excluded pending two conditions. First, days were excluded when confounding factors prevented system functioning resulting in only partial data collection, e.g. Wi-Fi outages. This occurred a total of 14 times. The microcontrollers used in this experiment rely on Wi-Fi connectivity to function normally. Wi-Fi signal in the greenhouse environment was very weak, as growing environments are not typically highly networked spaces. Second, days were excluded when the intensity threshold was so low that modulation control could not possibly meet the DLI target within 24 hours. This occurred 6 times at intensity thresholds less than or equal to 200 μmol· m^{-2}·s^{-1} for a DLI target of 17 mol· m^{-2}·d^{-1}. Days on which modulation control was deficit in meeting the DLI target were excluded from the sample because the intensity threshold is a hyperparameter to the algorithm, arbitrarily set by the user. This type of deficit is easily avoidable by growers who can set the intensity threshold based on the allowable photoperiod to ensure the DLI is met. The deficit does demonstrate the capacity of modulation control to precisely react to intensity settings determined by the grower, irrespective of the primary goal of meeting the DLI target. Low intensity thresholds may be advantageous given DLI targets lower than 17.0 mol· m^{-2}·d^{-1}, or for specialized applications like night break lighting (Runkle & Heins, 2005). After exclusion, there remained a representative sample of each intensity threshold (3-5 days) for which there was an even distribution of ambient light conditions (both cloudy and sunny days) within the typical DLI range for January in Ithaca (10-15 mol· m^{-2}·d^{-1}) (Korczynski et al., 2002).



	Binary	Modulation	Difference
Mean	17.054 ± 0.007	17.003 ± 0.002	0.051 ± 0.008
Standard Deviation	1.096e-02	4.012e-03	1.401e-02
Variance	1.201e-04	1.610e-05	1.963e-04

s

Figure 6

The difference in accuracy and precision between binary and modulation control methods is shown. Modulation control was more accurate and precise than binary control. Binary control exceeded the DLI target in all cases by a mean deviation of 0.05 mol· m^{-2}·d^{-1}, whereas modulation control exceeded the DLI target in most cases by a mean deviation of 0.003 mol· m^{-2}·d^{-1}. The respective means of the binary and modulation conditions are stated within a 95% confidence interval as 17.054 +- 0.007 and 17.003 +- 0.002. The mean difference between binary and modulation conditions is stated within a 95% confidence interval as 0.051 +- 0.0085.

Both modulation and binary controls met the DLI and intensity targets precisely and accurately; however, modulation control was more accurate and precise than binary control. Binary control exceeded the DLI target in all cases by a mean deviation of 0.05 mol· m^{-2}·d^{-1}, whereas modulation control exceeded the DLI target in most cases by a mean deviation of 0.003 mol· m^{-2}·d^{-1}. For a single day, modulation control was deficit in meeting the DLI target by 0.008 mol· m^{-2}·d^{-1}. The standard deviation and variance of modulation control were an order of magnitude lower than those of binary control. The respective means of the binary and modulation conditions are stated within a 95% confidence interval as 17.054 +- 0.007 and 17.003 +- 0.002. The mean difference between binary and modulation conditions is stated within a 95% confidence interval as 0.051 +- 0.0085. The entire interval of mean difference is positive and non-zero (0.0425, 0.0595), so a plausible difference in accuracy exists between the binary and modulation lighting control methods. Overall, both binary and modulation controls were very accurate in meeting the DLI target, and the difference between the two control methods represents very small differences in DLI.

Starting with the fundamental fixture efficacy (1.9 µmol / W) as reported by the manufacturer, the accuracy of modulation control can be expressed in economic terms via the efficiency of the fixture and the price of electricity (3) (4) (5).

$$(3)\ \frac{1.9\ \mu mols}{W} \times \frac{3600\ s}{h} \times \frac{kW}{1000\ W} = \frac{6.84\ mols}{kWh}$$

$$(4)\ \frac{mols}{m^2 d} \times \frac{kWh}{6.84\ mols} \times \frac{365\ d}{y} = \frac{kWh}{m^2 y}$$

$$(5)\ \frac{kWh}{m^2 y} \times \frac{\$}{kWh} = \frac{\$}{m^2 y}$$

The annual cost of precision of supplemental lighting would be $836.57 for a relatively small greenhouse of 4 hectares an average rate of $00.10 / kWh. The cost of precision of supplemental lighting would be $20,090.59 for a relatively large greenhouse of size 40 hectares, at a relatively expensive rate of $00.25 / kWh.

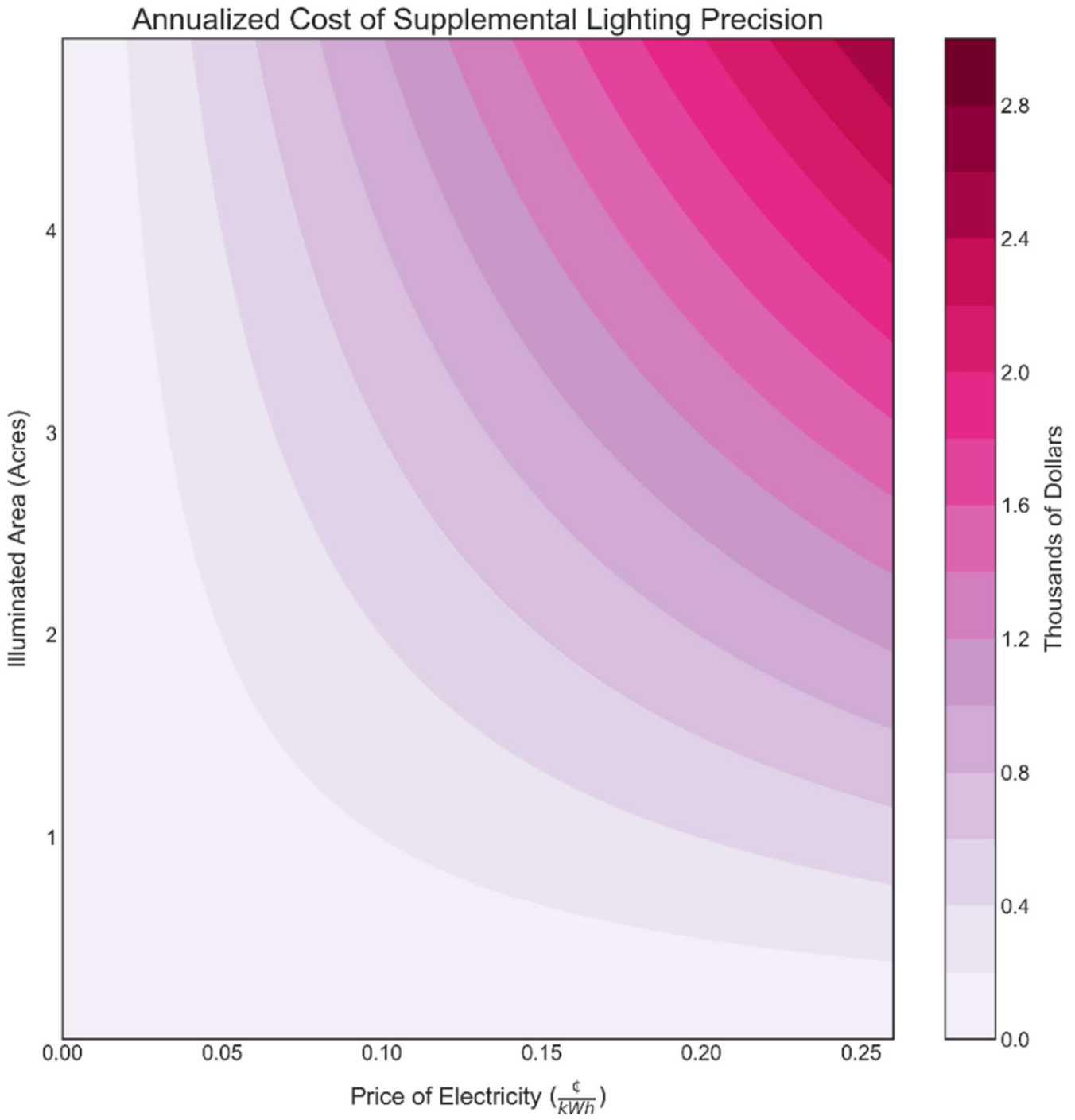


Figure 7

The annualized cost of precision of supplemental lighting control as it varies with the price of electricity and the area under cultivation. The annual cost of precision of supplemental lighting would be $836.57 for a relatively small greenhouse of 4 hectares an average rate of $00.10 / kWh. The cost of precision of supplemental lighting would be $20,090.59 for a relatively large greenhouse of size 40 hectares, at a relatively expensive rate of $00.25 / kWh.

Discussion

The cost of electricity motivates the precise and accurate control of supplemental lighting with respect to lighting targets. Assuming fixed electricity prices, low-intensity long duration lighting can be photosynthetically and electrically more efficient than high-intensity short duration (Zhen & van Iersel, 2017). However, variable electricity rates are commonplace, so dynamic management of intensity and photoperiod is necessary to avoid unreasonable costs. The off-peak electricity period, occurring from approximately midnight to dawn, offers significantly reduced electricity pricing. As utilities develop the smart-grid, dynamic energy management may offer new savings for growers (Clausen et al., 2015). Dynamic control strategies, that prioritize cheaper electricity while optimizing for energy when lighting must occur at peak rates, are necessary to carefully manage the cost of supplemental lighting. LASSI navigates the boundaries of the off-peak electricity period yet increasing the precision of lighting targets can further reduce the cost burden for growers. In the current experiment, the hanging height and fixture power determined the intensity of the binary control (approx. 700 μmol· m^{-2}·s^{-1}), which in this case exceeded the intensity threshold for all tested values (100 – 500 μmol· m^{-2}·s^{-1}). Compensating for increased intensity while striving to meet the DLI target, the binary control reacted with shorter photoperiods, which may have decreased the program's accuracy in meeting the DLI target. The modulation control had lower intensity supplemental lighting over longer photoperiods which gave the program more time to anticipate the DLI target.

The difference between the two control strategies was significant but also very small. Binary control met the target almost as well as modulation control. However, precision of supplemental lighting control does represent some amount of electricity cost to growers. The equations utilized here may facilitate comparison of different lighting control strategies in the future.

Networking Solutions for Growing Environments

Real-time analytics and actuation deployed in growing environments requires low latency networking solutions. Networked infrastructure may represent a continuous, cumulative cost as data products scale (Jamieson, 2017). Thus, low-cost hardware solutions of the kind employed in this research are necessary to translate evidence-based research into commercial settings. While in this case, network dependent functionality was a significant disadvantage, cloud connected microcontrollers represent the class of networked devices with the lowest cost. Running lighting control logic in firmware permits a degree of graceful degradation pending system failure. Integration of networked tools and services enables new features and increased computing power surpassing that of microcontrollers. Networked devices offer numerous advantages compared to embedded systems, including real-time analytic models, remote management, and software-as-a-service business models.

Sources of Error – Future Improvements

Deficiency in meeting the DLI target on the part of the modulation condition can be explained as an easily avoidable effect of the program's parameters. It was not physically possible for the program to provide enough supplemental light when constrained by low intensity targets and the absence of sunlight. Light gradients and irregular shading of the sensors by the greenhouse structure caused variation in concurrent conditions between treatments. Multiple sensors can be deployed to mitigate these effects. Modulation control still met low intensity targets with high accuracy. When combined with greater insolation, modulation control was able to meet the DLI target despite low intensity thresholds.

Binary control generally met the DLI target very precisely; however because the lamp was on at full power, PPFD under binary control was much higher than under modulation control. Depending on the crop, there may be diminished benefit to high intensity lighting against a high intensity ambient background. Modulation control generally responded with low intensity, long-duration supplemental lighting, which may provide added photosynthetic benefit.

There is much opportunity to combine different control regimes. Combining advanced knowledge of photoperiodic effects, photosynthetic efficiency as it responds to intensity, and spectral distribution as it affects plant morphology and carbon assimilation. Additionally, there is opportunity to pair lights with different sensors in biofeedback loops where biotic conditions of the crop are used to inform lighting decisions. When incorporating different control regimes, care must be taken to weight the inputs appropriately such that optimal cultivations conditions are maintained while reducing cost to the grower.

Conclusion

While increasing efficiencies in controlled environment crop production is tantamount to reducing the embodied energy that goes into food production, the Jevons' paradox complicates the role of technological development. For example, subsurface drip irrigation is an order of magnitude more efficient than flood irrigation; however, increased efficiency may spur cultivation of crops with a higher irrigation requirement (Sears, 2018). Technologies that increase energy efficiency must also be appropriate for a given context; energy efficient HVAC systems powered by fossil energy may not be appropriate in arid climates where passive evaporative cooling is cheap and efficient. Simple measures of efficiency bear no indication of the interaction a given technology has with the environment at large. Ecosystem level analysis is needed to weigh the costs

against the benefits of increasing efficiency in controlled environment agriculture. In this regard thermoeconomic analysis is useful.

Exergy is the maximum useful work occurring as systems interact to equilibrium (Bejan et al., 1999). A key component of exergetic analysis is that it accounts for interactions between systems and their environments, which contextualizes the utility of a technology relative to other technologies and systems. Non-obvious bottlenecks in energy flows emerge that may limit system functioning beyond pure thermodynamic optimization. Beyond simply minimizing entropy, thermoeconomics accounts for the destruction of exergy, or exergetic efficiency (Bejan et al., 1999). Exergetic efficiency informs the development of sustainable technologies that are appropriate for a given environment and adhere to basic physical and thermodynamic laws.

From an exergetic perspective, the direct use of sunlight is an agricultural imperative. More exergy is destroyed when using photovoltaics, batteries and luminaires built with precious metals sourced with fossil fuels. Fundamental efficiencies of each component drive energy conversion losses that limit the fundamental system efficacy. Nonetheless, the demand for uninterrupted energy services is high, i.e. the production of food. Solar energy is subject to latency, so an opportunity arises to mitigate its daily and annual flux. Supplemental lighting in greenhouses can extend the season while taking full advantage of the solar resource. Yields of high-light crops in the shoulder seasons, made possible with supplemental lighting, may prove essential towards increasing annualized yields in the context of fossil fuel scarcity. Similarly, utilities must navigate the daily flux in demand for power, and streetlighting at night can help to mitigate the cost of throttling supply (Haider, 2016). However, streetlighting at night represents a major energy expense that is largely underutilized and a major source of light pollution (Rodrigues et al., 2011). CEA facilities have an opportunity to engage utilities regarding demand-response and load-balancing of electricity production. Greenhouses growing day-neutral

crops are well positioned to preserve the exergy of electric lighting by producing food for human consumption instead of illuminating empty streets at night. There remains a dearth of research that applies exergetic analysis to the role of controlled environment agriculture in society.

References

1. Adams, S. R., & Langton, F. A. (2005). Photoperiod and plant growth: a review. The Journal of Horticultural Science and Biotechnology, 80(1), 2-10.
2. Albright, L. D., & Langhans, R. W. (1996). *Controlled Environment Agriculture Scoping Study*. Electric Power Research Institute.
3. Albright, L. D., Both, A. J., & Chiu, A. J. (2000). Controlling greenhouse light to a consistent daily integral. *Transactions of the ASAE*, *43*(2), 421.
4. Albright, L. D., de Villiers, D. S., & the Cornell Department of Biological and Environmental Engineering. (2008). Energy investments and CO2 emissions for fresh produce imported into New York State compared to the same crops grown locally. Final Report prepared for the New York State Energy Research and Development Authority, Cornell University, Ithaca, (NY), USA.
5. Annunziata, M. G., Apelt, F., Carillo, P., Krause, U., Feil, R., Mengin, V., ... & Lunn, J. E. (2017). Getting back to nature: a reality check for experiments in controlled environments. *Journal of experimental botany*, 68(16), 4463-4477.
6. Ayars, J. E., Fulton, A., & Taylor, B. (2015). Subsurface drip irrigation in California—Here to stay?. *Agricultural Water Management*, 157, 39-47.
7. Barnes, C., Tibbitts, T., Sager, J., Deitzer, G., Bubenheim, D., Koerner, G., & Bugbee, B. (1993). Accuracy of quantum sensors measuring yield photon flux and photosynthetic photon flux. *HortScience*, *28*(12), 1197-1200.
8. Bejan, A., Vadász, P., & Kröger, D. G. (Eds.). (2012). Energy and the Environment (Vol. 15). *Springer Science & Business Media.*

9. Benke, K., & Tomkins, B. (2017). Future food-production systems: vertical farming and controlled-environment agriculture. *Sustainability: Science, Practice and Policy*, 13(1), 13-26.

10. Both, A. J., Albright, L. D., Langhans, R. W., Reiser, R. A., & Vinzant, B. G. (1994, January). Hydroponic lettuce production influenced by integrated supplemental light levels in a controlled environment agriculture facility: experimental results. In *III International Symposium on Artificial Lighting in Horticulture 418* (pp. 45-52).

11. Both, A. J., Bugbee, B., Kubota, C., Lopez, R. G., Mitchell, C., Runkle, E. S., & Wallace, C. (2017). Proposed Product Label for Electric Lamps Used in the Plant Sciences. *HortTechnology*, *27*(4), 544

12. Brown, C. S., Schuerger, A. C., & Sager, J. C. (1995). Growth and photomorphogenesis of pepper plants under red light-emitting diodes with supplemental blue or far-red lighting. Journal of the American Society for Horticultural Science, 120(5), 808-813.

13. Bruinsma, J. (2017). World agriculture: towards 2015/2030: an FAO study. Routledge. Bruinsma, J. (2017). *World agriculture: towards 2015/2030: an FAO study*. Routledge.

14. Brundtland, G. H., Khalid, M., Agnelli, S., & Al-Athel, S. (1987). Our common future. New York.

15. Bugbee, B. (2016). Toward an optimal spectral quality for plant growth and development: the importance of radiation capture, 1–12. https://doi.org/10.17660/ActaHortic.2016.1134.1

16. Bugbee, B. G., & Salisbury, F. B. (1988). Exploring the limits of crop productivity: I. Photosynthetic efficiency of wheat in high irradiance environments. Plant Physiology, 88(3), 869-878.

17. Bula, R. J., Tennessen, D. J., Morrow, R. C., & Tibbitts, T. W. (1994). Light emitting diodes as a plant lighting source.

18. Carlsson-Kanyama, A., Ekström, M. P., & Shanahan, H. (2003). Food and life cycle energy inputs: consequences of diet and ways to increase efficiency. Ecological economics, 44(2-3), 293-307.

19. Challa, H., Nederhoff, E. M., Bot, G. P. A., & van de Braak, N. J. (1988, May). Greenhouse climate control in the nineties. In Symposium on High Technology in Protected Cultivation 230 (pp. 459-470).

20. Ciolkosz, D. (2008). Design daylight availability for greenhouses using supplementary lighting. *Biosystems engineering*, *100*(4), 571-580.

21. Clausen, A., Maersk-Moeller, H. M., Soerensen, J. C., Joergensen, B. N., Kjaer, K. H., & Ottosen, C. O. (2015). Integrating commercial greenhouses in the smart grid with demand response based control of supplemental lighting. In *International Conference Industrial Technology Management Science (ITMS 2015)* (pp. 199-213).

22. Cockshull, K. E., Graves, C. J., & Cave, C. R. (1992). The influence of shading on yield of glasshouse tomatoes. *Journal of Horticultural Science*,*67*(1), 11-24.

23. Cope, K. R., Snowden, M. C., & Bugbee, B. (2014). Photobiological Interactions of Blue Light and Photosynthetic Photon Flux : Effects of Monochromatic and Broad-Spectrum Light Sources, 574–584. https://doi.org/10.1111/php.12233

24. Craford, M. G., Shaw, R. W., Herzog, A. H., & Groves, W. O. (1972). Radiative recombination mechanisms in GaAsP diodes with and without nitrogen doping. *Journal of Applied Physics*, 43(10), 4075-4083.

25. Datt, P. (2011). Latent heat of vaporization/condensation. *Encyclopedia of snow, ice and glaciers*, 703-703.

26. De Villiers, D. S., Wien, H. C., Reid, J. E., & Albright, L. D. (2009, June). Energy use and yields in tomato production: field, high tunnel and greenhouse compared for the northern tier of the USA (Upstate New York). In International Symposium on High Technology for Greenhouse Systems: GreenSys2009 893 (pp. 373-380).

27. Dueck, T. A., Janse, J., Eveleens, B. A., Kempkes, F. L. K., & Marcelis, L. F. M. (2011, June). Growth of tomatoes under hybrid LED and HPS lighting. In International Symposium on Advanced Technologies and Management Towards Sustainable Greenhouse Ecosystems: Greensys2011 952 (pp. 335-342).

28. Duffy, S., 2018. Dynamic Lighting Controls for Greenhouses. Retrieved from: https://duffy-thesis-cornell.herokuapp.com

29. Food and Agriculture Organization of the United Nations (FAO). (2011). Energy-Smart Food for People Climate, 66. https://doi.org/2/3/2017

30. Frantz, J. M., Joly, R. J., & Mitchell, C. A. (2000). Intracanopy lighting influences radiation capture, productivity, and leaf senescence in cowpea canopies. Journal of the American Society for Horticultural Science, 125(6), 694-701.

31. Frantz, J. M., Ritchie, G., Cometti, N. N., Robinson, J., & Bugbee, B. (2004). Exploring the limits of crop productivity: beyond the limits of tipburn in lettuce. Journal of the American Society for Horticultural Science, 129(3), 331-338.

32. Frenken, K., & Gillet, V. (2012). Irrigation water requirement and water withdrawal by country. FAO, Rome, Italy.

33. Fridley, D. (2010). Nine challenges of alternative energy. The post carbon reader: managing the, 21.

34. Graamans, L., Baeza, E., Van Den Dobbelsteen, A., Tsafaras, I., & Stanghellini, C. (2018). Plant factories versus greenhouses: Comparison of resource use efficiency. *Agricultural Systems*, 160, 31-43.

35. Groves, W. O., Herzog, A. H., & Craford, M. G. (1971). The Effect of Nitrogen Doping on GaAs1− x P x Electroluminescent Diodes. *Applied Physics Letters*, 19(6), 184-186.

36. Gunnlaugsson, B., & Adalsteinsson, S. (2005, June). Interlight and plant density in year-round production of tomato at northern latitudes. In V International Symposium on Artificial Lighting in Horticulture 711 (pp. 71-76).

37. Gupta, S. D., & Dutta Agarwal, A. (2017). Light Emitting Diodes for Agriculture. Springer: Singapore.

38. Haider, H. T., See, O. H., & Elmenreich, W. (2016). A review of residential demand response of smart grid. *Renewable and Sustainable Energy Reviews*, 59, 166-178.

39. Haitz, R., & Tsao, J. Y. (2011). Solid-state lighting: 'The case'10 years after and future prospects. *physica status solidi* (a), 208(1), 17-29.

40. Harbick, K., & Albright, L. D. (2016, May). Comparison of energy consumption: greenhouses and plant factories. In *VIII International Symposium on Light in Horticulture 1134* (pp. 285-292).

41. Harbick, K., Albright, L. D., & Mattson, N. S. (2016). Electrical savings comparison of supplemental lighting control systems in greenhouse environments. In 2016 ASABE Annual International Meeting (p. 1). American Society of Agricultural and Biological Engineers.

42. Hardin, G. (1968). The tragedy of the commons. science, 162(3859), 1243-1248.

43. Hernández, R., & Kubota, C. (2015). Physiological, morphological, and energy-use efficiency comparisons of LED and HPS supplemental lighting for cucumber transplant production. *HortScience*, *50*(3), 351-357.

44. Heuvelink, E. P., & Challa, H. (1989, May). Dynamic optimization of artificial lighting in greenhouses. In *International Symposium on Growth and Yield Control in Vegetable Production 260* (pp. 401-412).

45. Heuvelink, E., Bakker, M. J., Hogendonk, L., Janse, J., Kaarsemaker, R., & Maaswinkel, R. (2005, June). Horticultural lighting in the Netherlands: new developments. In *V International Symposium on Artificial Lighting in Horticulture 711* (pp. 25-34).

46. Holonyak Jr, N., & Bevacqua, S. F. (1962). Coherent (visible) light emission from Ga (As1− x P x) junctions. *Applied Physics Letters*, 1(4), 82-83.

47. Islam, M. A., Kuwar, G., Clarke, J. L., Blystad, D. R., Gislerød, H. R., Olsen, J. E., & Torre, S. (2012). Artificial light from light emitting diodes (LEDs) with a high portion of blue light results in shorter poinsettias compared to high pressure sodium (HPS) lamps. *Scientia Horticulturae*, 147, 136-143.

48. Jamieson, D. (2017). The Emerging Economics of IoT. Particle.

49. Kim, H. H., Goins, G. D., Wheeler, R. M., & Sager, J. C. (2004). Green-light supplementation for enhanced lettuce growth under red-and blue-light-emitting diodes. *HortScience*, *39*(7), 1617-1622.

50. Kjaer, K. H., Ottosen, C. O., & Jørgensen, B. N. (2011). Cost-efficient light control for production of two campanula species. *Scientia horticulturae*, *129*(4), 825-831.

51. Korczynski, P. C., Logan, J., & Faust, J. E. (2002). Mapping monthly distribution of daily light integrals across the contiguous United States. *HortTechnology*, 12(1), 12-16.

52. Kubota, C., Kroggel, M., Both, A. J., Burr, J. F., & Whalen, M. (2016, May). Does supplemental lighting make sense for my crop?-empirical evaluations. In *VIII International Symposium on Light in Horticulture 1134* (pp. 403-412).

53. Kubota, C., Kroggel, M., Both, A. J., Burr, J. F., & Whalen, M. (2016, May). Does supplemental lighting make sense for my crop?-empirical evaluations. In *VIII International Symposium on Light in Horticulture 1134* (pp. 403-412).

54. Levine, M., Ürge-Vorsatz, D., Blok, K., Geng, L., Harvey, D., Lang, S., ... & Rilling, J. (2007). Residential and commercial buildings. Climate change, 20, 17.

55. Losev, O. V. (1927). BLuminous carborundum detector and detection with crystals,[Telegrafiya i Telefoniya bez Provodov, vol. 44.

56. Marcelis, L. F. M., Broekhuijsen, A. G. M., Meinen, E., Nijs, E. M. F. M., & Raaphorst, M. G. M. (2005, June). Quantification of the growth response to light quantity of greenhouse grown crops. In *V International Symposium on Artificial Lighting in Horticulture 711* (pp. 97-104).

57. Martineau, V., Lefsrud, M., Naznin, M. T., & Kopsell, D. A. (2012). Comparison of light-emitting diode and high-pressure sodium light treatments for hydroponics growth of Boston lettuce. HortScience, 47(4), 477-482.

58. Massa, G. D., Kim, H. H., Wheeler, R. M., & Mitchell, C. A. (2008). Plant productivity in response to LED lighting. HortScience, 43(7), 1951-1956.

59. Narumoto, M., Cruz, A., Bowen, R., Wasson, M., Bennage, C. (2018). API Design. Microsoft Azure. Retrieved from: https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design

60. Nelson, J. A., & Bugbee, B. (2014). Economic analysis of greenhouse lighting: light emitting diodes vs. high intensity discharge fixtures. *PloS one*,*9*(6), e99010.

61. Nelson, J. A., & Bugbee, B. (2015). Analysis of environmental effects on leaf temperature under sunlight, high pressure sodium and light emitting diodes. *PloS one*, *10*(10), e0138930.

62. Orr, D. W. (1992). Ecological literacy: Education and the transition to a postmodern world. Suny Press.

63. Ostrom, E. (2002). Agriculture and its External Linkages. Handbook of Agricultural Economics (Vol. 2). Elsevier. https://doi.org/10.1016/S1574-0072(02)10006-5

64. Pattison, P. M., Tsao, J. Y., Brainard, G. C., & Bugbee, B. (2018). LEDs for photons, physiology and food. Nature, 563(7732), 493.

65. Pinho, P., Hytönen, T., Rantanen, M., Elomaa, P., & Halonen, L. (2013). Dynamic control of supplemental lighting intensity in a greenhouse environment. Lighting Research & Technology, 45(3), 295-304.

66. Poorter, H., Fiorani, F., Pieruschka, R., Wojciechowski, T., Putten, W. H., Kleyer, M., ... & Postma, J. (2016). Pampered inside, pestered outside? Differences and similarities between plants growing in controlled conditions and in the field. New Phytologist, 212(4), 838-855.

67. Poulet, L., Massa, G. D., Morrow, R. C., Bourget, C. M., Wheeler, R. M., & Mitchell, C. A. (2014). Significant reduction in energy for plant-growth lighting in space using targeted LED lighting and spectral manipulation. *Life Sciences in Space Research*, *2*, 43-53.

68. Pust, P., Schmidt, P. J., & Schnick, W. (2015). A revolution in lighting. *Nature materials*, 14(5), 454.

69. Rodrigues, C. R., Almeida, P. S., Soares, G. M., Jorge, J. M., Pinto, D. P., & Braga, H. A. (2011, June). An experimental comparison between different technologies arising for public lighting: LED luminaires replacing high pressure sodium lamps. In 2011 IEEE *International Symposium on Industrial Electronics* (pp. 141-146). IEEE.

70. Round, H. J. (1991). A note on carborundum. In Semiconductor Devices*: Pioneering Papers* (pp. 879-879).

71. Runkle, E. S., & Heins, R. D. (2005, June). Manipulating the light environment to control flowering and morphogenesis of herbaceous plants. In *V International Symposium on Artificial Lighting in Horticulture 711* (pp. 51-60).

72. Sager, J. C., Smith, W. O., Edwards, J. L., & Cyr, K. L. (1988). Photosynthetic efficiency and phytochrome photoequilibria determination using spectral data. Transactions of the ASAE, 31(6), 1882-1889.

73. Sears, L., Caparelli, J., Lee, C., Pan, D., Strandberg, G., Vuu, L., & Lin Lawell, C. Y. (2018). Jevons' Paradox and efficient irrigation technology. *Sustainability*, 10(5), 1590.

74. Seginer, I., Ioslovich, I., & Albright, L. D. (2005). Strategies for a Constant Daily Light Integral in Greenhouses. *Acta horticulturae*, *691*(1), 117.

75. Snowden, M. C., Cope, K. R., & Bugbee, B. (2016). Sensitivity of seven diverse species to blue and green light: interactions with photon flux. *PloS one*, *11*(10), e0163121.

76. Steffen, W., Richardson, K., Rockström, J., Cornell, S. E., Fetzer, I., Bennett, E. M., ... & Folke, C. (2015). Planetary boundaries: Guiding human development on a changing planet. *Science*, *347*(6223), 1259855.

77. Talens, P., Villalba, M., Sciubba, E., & Gabarrell i Durany, X. (2010). Extended exergy accounting applied to biodiesel production. *Energy (Oxford),* 35(7), 2861-2869.

78. Tennessen, D. J., Bula, R. J., & Sharkey, T. D. (1995). Efficiency of photosynthesis in continuous and pulsed light emitting diode irradiation. *Photosynthesis research*, *44*(3), 261-269.

79. Terashima, I., Fujita, T., Inoue, T., Chow, W. S., & Oguchi, R. (2009). Green light drives leaf photosynthesis more efficiently than red light in strong white light: revisiting the enigmatic question of why leaves are green. *Plant and cell physiology*, *50*(4), 684-697.

80. Thimijan, R. W., & Heins, R. D. (1983). Photometric , Radiometric , and Quantum Light Units of Measure A Review of Procedures for Interconversion, 18(December).

81. Tsao, J. Y., Han, J., Haitz, R. H., & Pattison, P. M. (2015). The Blue LED Nobel Prize: Historical context, current scientific understanding, human benefit. Annalen der Physik, 527(5-6), A53-A61.

82. United Nations, Department of Economic and Social Affairs, Population Division (2019). World Population Prospects 2019: Ten Key Findings.

83. Van Iersel, M. W. (2003). Carbon use efficiency depends on growth respiration, maintenance respiration, and relative growth rate. A case study with lettuce. Plant, Cell & Environment, 26(9), 1441-1449.

84. van Iersel, M. W., & Gianino, D. (2017). An Adaptive Control Approach for Light-emitting Diode Lights Can Reduce the Energy Costs of Supplemental Lighting in Greenhouses. HortScience, 52(1), 72-77.

85. van Iersel, M. W., Weaver, G., Martin, M. T., Ferrarezi, R. S., Mattos, E., & Haidekker, M. (2016). A chlorophyll fluorescence-based biofeedback system to control photosynthetic lighting in controlled environment agriculture. *Journal of the American Society for Horticultural Science*, *141*(2), 169-176.

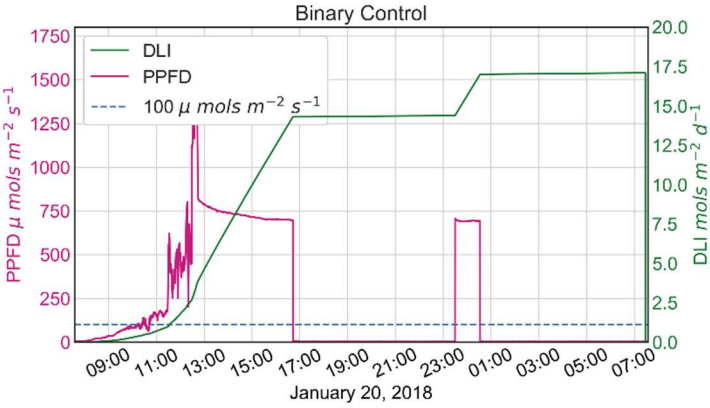
86. Vivanco, D. F., McDowall, W., Freire-González, J., Kemp, R., & van der Voet, E. (2016). The foundations of the environmental rebound effect and its contribution towards a general framework. *Ecological Economics*, 125, 60-69.

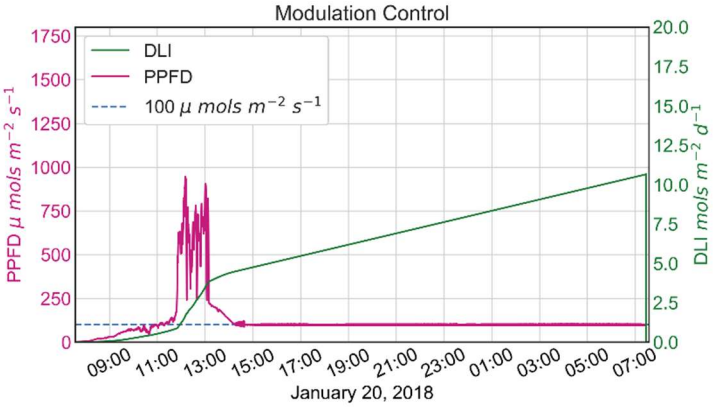
87. Weber, C. L., & Matthews, H. S. (2008). Food-miles and the relative climate impacts of food choices in the United States.Zeidler, C., Schubert, D., & Vrakking, V. (2013). Feasibility study: vertical farm EDEN (Doctoral dissertation, DLR Institute of Space Systems).

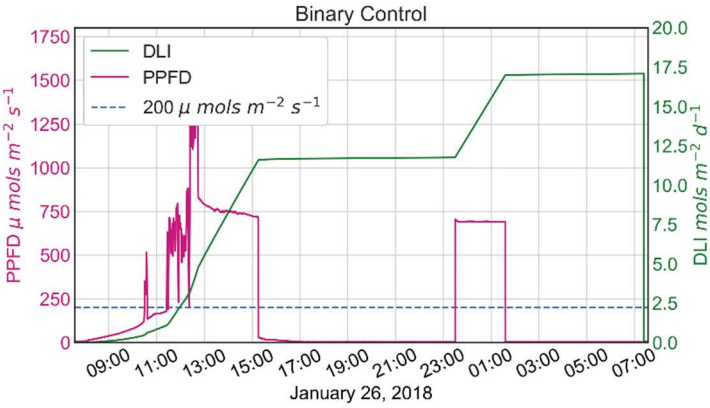
88. Zeidler, C., Schubert, D., & Vrakking, V. (2013). Feasibility study: vertical farm EDEN (Doctoral dissertation, DLR Institute of Space Systems).

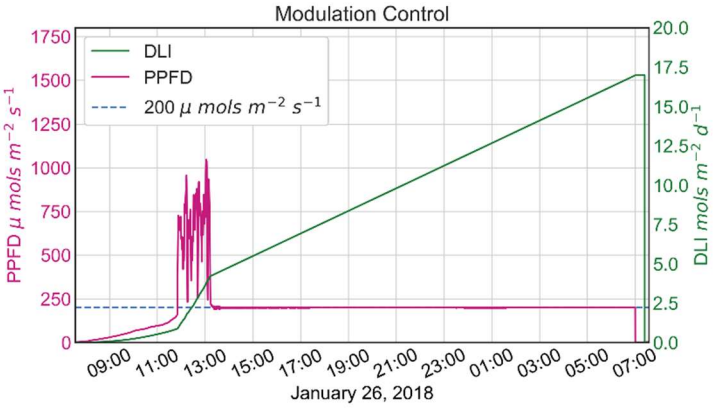
Appendix

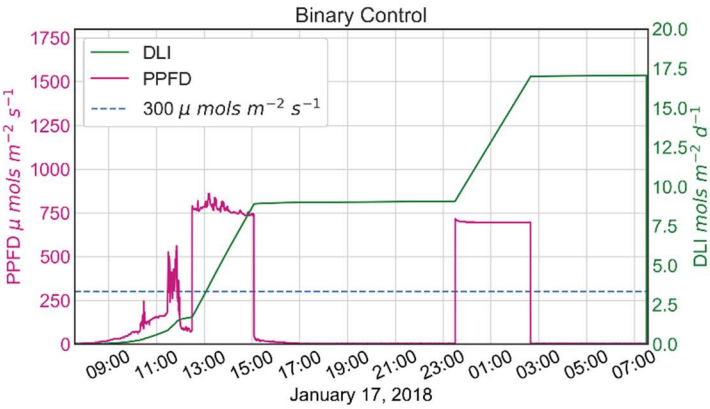
Daily Light Measurements

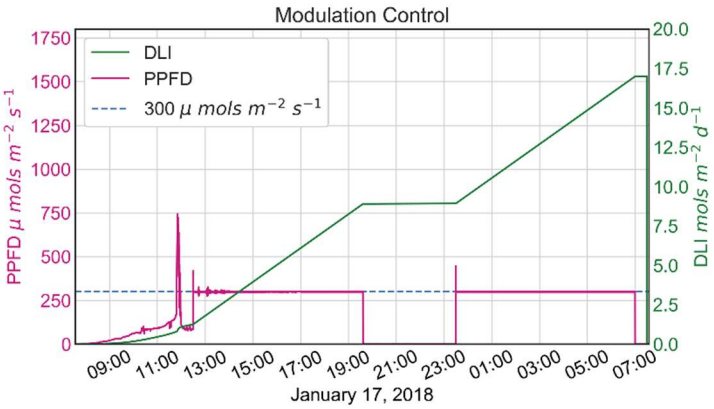


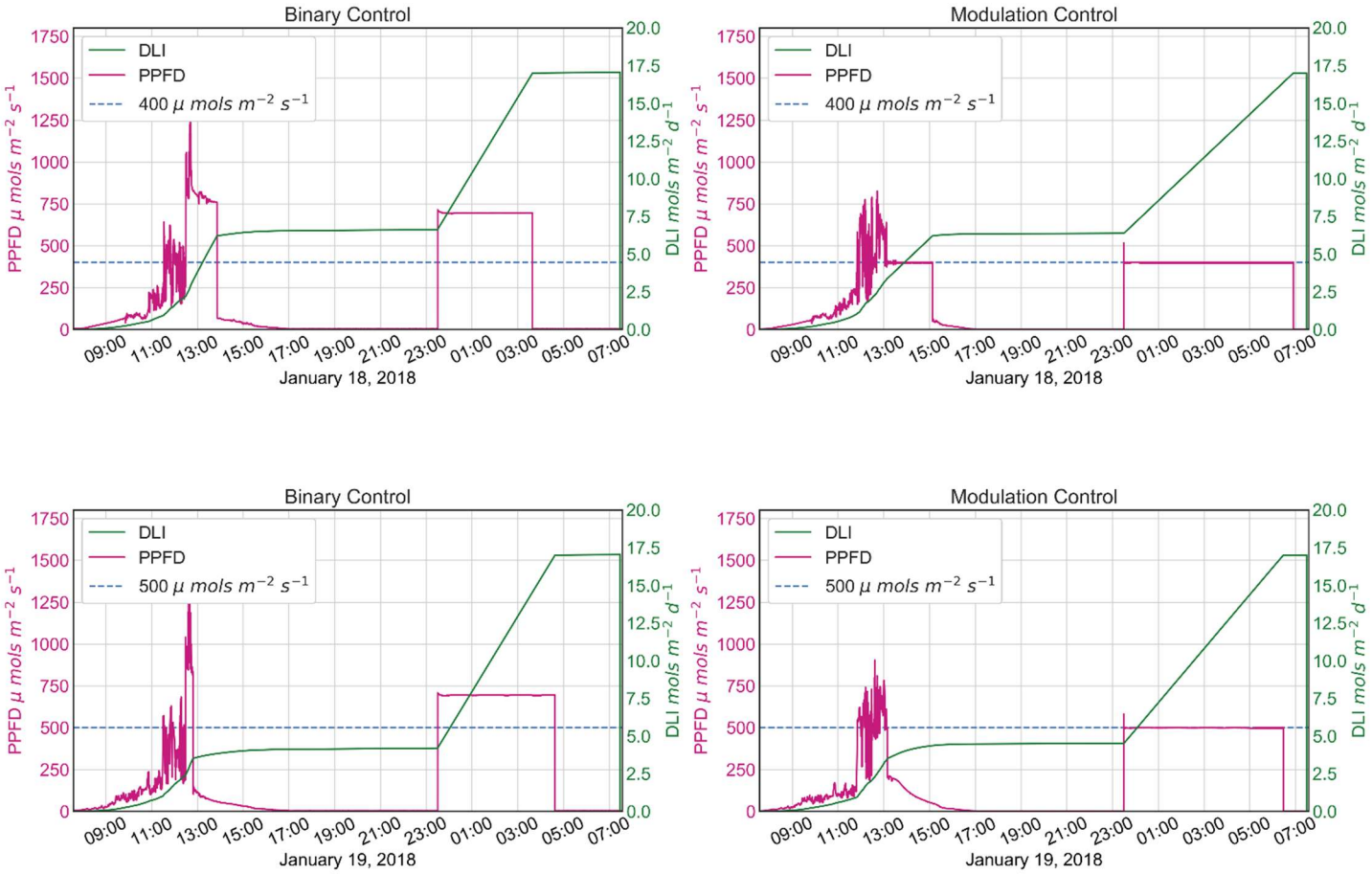




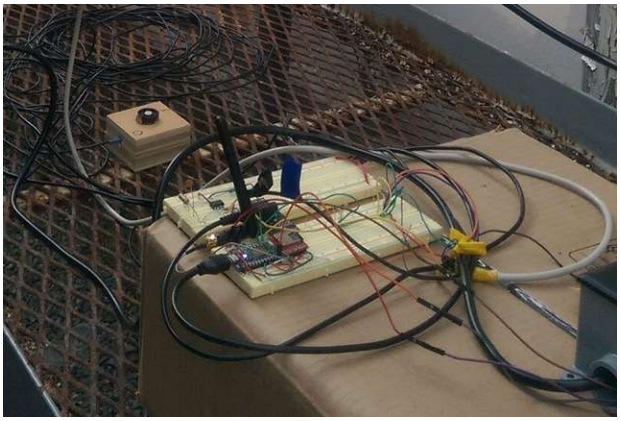


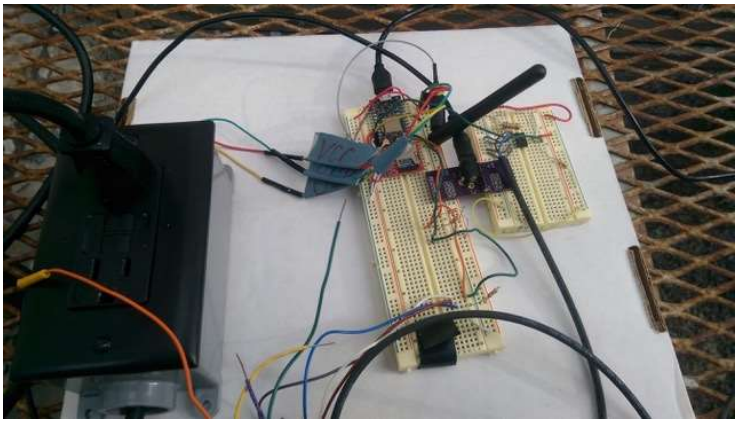





Binary Control
Modulation Control
DLI
PPFD
400 μ mols m⁻² s⁻¹
500 μ mols m⁻² s⁻¹
PPFD μ mols m⁻² s⁻¹
DLI mols m⁻² d⁻¹
0
250
500
750
1000
1250
1500
1750
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
09:00
11:00
13:00
15:00
17:00
19:00
21:00
23:00
01:00
03:00
05:00
07:00
January 18, 2018
January 19, 2018

Photos of Experimental Setup





Lighting and Shading System Implementation Code (See next page)

```
1
2 #include <math.h>
3 #include "SparkJson.h"
4 #include "Sunrise.h"
5
6 //=============================================BEGIN SDFAT SETUP
7 #include "SdFat.h"
8 // Pick an SPI configuration.
9 // See SPI configuration section below (comments are for photon).
10 #define SPI_CONFIGURATION 0
11 //------------------------------------------------------------------------------
12 // Setup SPI configuration.
13 #if SPI_CONFIGURATION == 0
14 // Primary SPI with DMA
15 // SCK => A3, MISO => A4, MOSI => A5, SS => A2 (default)
16 SdFat sd;
17 const uint8_t chipSelect = SS;
18 #elif SPI_CONFIGURATION == 1
19 // Secondary SPI with DMA
20 // SCK => D4, MISO => D3, MOSI => D2, SS => D1
21 SdFat sd(1);
22 const uint8_t chipSelect = D1;
23 #elif SPI_CONFIGURATION == 2
24 // Primary SPI with Arduino SPI library style byte I/O.
25 // SCK => A3, MISO => A4, MOSI => A5, SS => A2 (default)
26 SdFatLibSpi sd;
27 const uint8_t chipSelect = SS;
28 #elif SPI_CONFIGURATION == 3
29 // Software SPI.  Use any digital pins.
30 // MISO => D5, MOSI => D6, SCK => D7, SS => D0
31 SdFatSoftSpi<D5, D6, D7> sd;
32 const uint8_t chipSelect = D0;
33 #endif  // SPI_CONFIGURATION
34 //------------------------------------------------------------------------------
35 File myFile;
36 //=============================================END SDFAT SETUP
37
38 //=============================================begin i2c SETUP
39 const byte address = 0b0101100;
40
41 const byte wiperRedR = 0b00011100;
42 const byte wiperRedW = 0b00010000;
43 const byte wiperRedI = 0b00010100;
44 const byte wiperRedD = 0b00011000;
45
46 const byte wiperBlueR = 0b01101100;
47 const byte wiperBlueW = 0b01100000;
48 const byte wiperBlueI = 0b01100100;
49 const byte wiperBlueD = 0b01101000;
50
51 const byte wiperWhiteR = 0b01111100;
52 const byte wiperWhiteW = 0b01110000;
53 const byte wiperWhiteI = 0b01110100;
54 const byte wiperWhiteD = 0b01111000;
55
56 bool flip = true;
57
58
59 //=============================================end i2c SETUP
60
61 //=============================================CUSTOM VARIABLES
62
63 TCPClient client;
64
65 enum State {
66       START,
67       STATE_CONNECT,
```

```
68       STATE_READ,
69       STATE_PRODUCTION,
70       STATE_DISCONNECT,
71       START_SOCKET,
72       STATE_CONNECT_SOCKET,
73       STATE_READ_SOCKET,
74       STATE_WRITE_SOCKET,
75       SDFAT_WRITE,
76       SDFAT_READ,
77       TEST
78       };
79
80 State state = STATE_PRODUCTION;
81 //=============================================================READ VARIABLES
82 //======INDEX FOR INCOMING PACKETS
83 int day = 1;
84 String year = "2015";
85
86 bool tcp = false;
87 bool tmy = false;
88 bool socket = false;
89 bool production = true;
90
91 char inChar[2000];
92 int j = 0;
93 bool record = false;
94 //=============================================================TRANSMISSION VARS
95 // char server[] = "ec2-34-209-90-103.us-west-2.compute.amazonaws.com";
96 // int port = 8080;
97
98
99
100 // char server[] = "duffy-thesis-cornell-mod.herokuapp.com";
101 // int port = 80;
102
103
104 char server[] = "duffy-thesis-cornell-binary.herokuapp.com";
105 int port = 80;
106
107
108 //=============================================================set to true for binary control -- else PWM
109 bool binary = true;
110
111 //=================================================TIME CLOCK CONTROL VARIABLES
112 bool timeClockOn = false;
113 unsigned int timeClockStart = 7*3600000;
114 unsigned int timeClockFinish = 23*3600000;
115 //=================================================PROGRAM START SECONDS
116 unsigned int seconds_start = 0;
117 unsigned int second_counter = 0;
118 float filtered_signal_previous = 0.0;
119
120 //=============================================================MISC VARIABLES
121 int rule_flag = 0;
122 String myRandWebSocket = String(rand()*10000+10000); //attempt at random security
123 char buffer[4000];
124 char SdFatBuffer[4000];
125 bool init = true;
126 bool once = true;
127 bool SdFatOnce = true;
128 //=============================================================PWM CONTROL SIGNAL
129
130 const int licor_sensor_pin = A1;
131 const int energy_pin = A0;
132
133 //=============================================================DLI RESET
134 float DLItarget = 17.0;
135 float DLI_counter = 0.0;
136
137 float PPFD_full_power = 550.0;
```

```
138 float PPFD_target = 500.00;
139 float photoperiod = 24;
140
141 //================================================ POWER METER INIT
142 const int vin_pin = 1;
143 const int numSamples = 100;
144 float Vref = 1570; // Output voltage with no current: ~ 1570 mV or 1.57V
145 float sensitivity = 5400.0 / 1570.0; //5400mA per 1570mv = 3.43
146
147 int setPPFDtarget(String command);
148
149
150 void setup() {
151
152   WiFi.selectAntenna(ANT_EXTERNAL);
153
154   Wire.setSpeed(CLOCK_SPEED_100KHZ);
155   Wire.stretchClock(false);
156   Wire.begin();
157
158   Serial.begin(9600);
159   if (tcp) {
160     Time.zone(0);
161
162   } else {
163     Time.zone(-5);
164
165   }
166
167   filtered_signal_previous = licor_sensor_PPFD();
168   // initialize transmission JSON
169   seconds_start = Time.local();
170   second_counter = Time.now();
171 //=============================================SETUP SDFAT
172   if (!sd.begin(chipSelect, SPI_HALF_SPEED)) {
173     Serial.println("Initialization of SdFat failed. Stopping...");
174   //  sd.initErrorHalt();
175   }
176 //============================================================PWM control init
177   // float PPFD_actual = readPPFD();
178
179   // PWM_min = 4095.0*0.05;
180   // PWM_max = 4095.0*0.4;
181   //
182   // float PPFD_target = _PPFD_target(photoperiod, DLItarget);
183   // float difference = constrain((PPFD_target - PPFD_actual),0.0,PPFD_target);
184   // PWM_control = float_map(difference,0.0,(float)PPFD_full_power,(float)PWM_min,(float)PWM_max);
185   Particle.function("brew", setPPFDtarget);
186
187 }
188
189 int setPPFDtarget(String command)
190 {
191   // look for the matching argument "coffee" <-- max of 64 characters long
192   if(command == "500" || command == "400" || command == "300" || command == "200" || command == "100"
193   {
194     PPFD_target = atoi(command);
195     return atoi(command);
196
197   }
198   else return -1;
199 }
200
201 void GET(){
202
203   String file = String("000"+String(day));
204   String _file = file.substring(file.length()-3);
205   Serial.println(_file);
206   Serial.println("connected GET");
207   client.println("GET /api/firmware/day/"+year+"/"+_file+" HTTP/1.1");
```

```
208   client.println("Host: "+String(server));
209   client.println("Content-Length: 0");
210   client.println(); // send msg
211   day++;
212   delay(2000);
213 
214 }
215 
216 void POST(){
217 
218   int len = strlen(buffer);
219 
220   if (client.connect(server, port)){;
221     Serial.println();
222     Serial.println("Connected POST");
223     client.println("POST /api/firmware/datalogger/"+year+" HTTP/1.1");
224     client.println("Host: "+String(server));
225     client.printlnf("Content-Length: %d", len);
226     client.println("Content-Type: application/json; charset=UTF-8");
227     client.println();
228     client.println(buffer);
229     client.flush();
230     client.stop();
231     buff_reset();
232     delay(1000);
233 
234     } else {
235 
236       Serial.println("not connected");
237 
238     }
239 
240 }
241 
242 void GET_SOCKET(char route[]){
243 
244   client.write("GET "+String(route)+" HTTP/1.1\r\n");
245   client.write("Host: "+String(server)+"\r\n");
246   client.write("Upgrade: websocket\r\n");
247   client.write("Connection: Upgrade\r\n");
248   client.write("Sec-WebSocket-Key: "+String(myRandWebSocket)+"==\r\n");
249   client.write("Origin: photon\r\n");
250   client.write("Sec-WebSocket-Version: 13\r\n");
251   client.write("\r\n");
252 
253 }
254 
255 void POST_SOCKET(){
256 
257   uint8_t len = strlen(buffer);
258   int frameLength = 2;
259   byte first = byte(129);
260   byte second = byte(len);
261   byte third = byte((len >> 8) & 255);
262   byte fourth = byte(len & 255);
263 
264   if (len > 125){
265     second = byte(126);
266     frameLength = 4;
267   }
268 
269   byte frame[frameLength];
270   frame[0] = first;
271   frame[1] = second;
272 
273   if (len > 125){
274     frame[2] = third;
275     frame[3] = fourth;
276   }
277 
```

```
278   client.write(frame,frameLength);
279   int written = client.write(buffer);
280   buff_reset();
281
282
283 }
284 void _read(){
285
286   //  Serial.println("client available");
287   char c = client.read();
288   Serial.print(c);
289   if (c == '[') {record = true; }
290
291   if (record){
292     inChar[j] = c;
293     j++;
294 //    if (c == ']') {record = false; }
295   }
296 }
297
298 void _read_reset(){
299   //=========================================================REINITIALIZE READ VARIABLES
300   j=0;
301   record = false;
302   for (int k = 0; k<sizeof(inChar); k++){
303     inChar[k]='\0';
304   };
305 }
306
307 void buff_reset(){
308
309   for (int k = 0; k<sizeof(buffer); k++){
310     buffer[k]='\0';
311   };
312 }
313
314 void SdFat_reset(){
315
316   for (int k = 0; k<sizeof(SdFatBuffer); k++){
317     SdFatBuffer[k]='\0';
318   };
319
320 }
321
322 //=========================================================================LOOP
323 void loop(){
324
325   switch(state) {
326 //=============================================================DISCONNECT
327       case STATE_DISCONNECT:
328         client.stop();
329         for(;;);
330       break;
331
332 //=================================================================START
333       case START:
334         while(!Serial.available()) Particle.process();
335         state = STATE_CONNECT;
336       break;
337
338 //=================================================================START
339       case START_SOCKET:
340         while(!Serial.available()) Particle.process();
341         state = STATE_CONNECT_SOCKET;
342       break;
343
344 //=========================================================SOCKET CONNECT
345       case STATE_CONNECT_SOCKET:
346
347         Serial.println();
```

```
348         Serial.println("STATE: SOCKET_CONNECT");
349 
350         if (client.connect(server, port)){
351 
352           Serial.println();
353           Serial.println("GET SOCKET");
354 
355           GET_SOCKET("/api/firmware/socket");
356           delay(2000);
357           state = STATE_READ_SOCKET;
358 
359         } else {
360 
361           delay(1000);
362           Serial.println("ERROR! Failed to connect! Retry...");
363 
364         }
365 
366       break;
367 
368 //============================================================READ SOCKET
369       case STATE_READ_SOCKET:
370 
371         if (!client.connected()){
372           Serial.println();
373           Serial.println("STATE: READ SOCKET -- Client Disconnected!");
374           state = STATE_CONNECT_SOCKET;
375           init = true;
376 
377         } else {
378 
379         //  Serial.println();
380         //  Serial.println("STATE: READ SOCKET -- Client UN-Available");
381 
382           if (client.available()){
383 
384             Serial.println();
385             Serial.println("STATE: READ SOCKET -- Client Available");
386 
387 
388             while(client.available()){
389               _read();
390             }
391 
392             client.flush();
393 
394           }
395 
396           state = STATE_WRITE_SOCKET;
397 
398         }
399 
400       break;
401 //=========================================================STATE WRITE SOCKET
402       case STATE_WRITE_SOCKET:
403 
404         if (strlen(inChar)>0){
405 
406           Serial.println();
407           Serial.println();
408           Serial.println("STATE: WRITE SOCKET -- SDFAT -- POST -- SOCKET");
409           Serial.println();
410 
411           // Serial.print("State before wrapper: ");
412           // Serial.println(state);
413 
414           if (production) {
415             Serial.println("--------------------------RAN: WRAPPER_PRODUCTION");
416             Serial.println();
417             wrapper_production();
```

```
418           } else {
419             Serial.println("------------------------------RAN: WRAPPER_SOCKET");
420             Serial.println();
421             // wrapper_socket();
422           }
423 
424           if (SdFatOnce){
425             //==============skip duplicate write to SdFat on init
426             SdFatOnce = false;
427           } else {
428             SdFatWrite(SdFatBuffer);
429           }
430           //SdFatRead();
431           POST_SOCKET();
432           _read_reset();
433           state = STATE_READ_SOCKET;
434 
435         } else {
436 
437 //============================================================GET LAST DOC IN DATABASE or create first
438           if (init){
439             init = false;
440             Serial.println();
441             Serial.println();
442             Serial.println("STATE: WRITE SOCKET -- get init -- ");
443             Serial.println();
444 
445             String _init = "init";
446             _init.toCharArray(buffer,sizeof(buffer));
447             POST_SOCKET();
448 
449           }
450 
451 //================================================if statement executes once per second
452           int now = Time.now();
453           if (now != second_counter){
454             second_counter = now;
455             unsigned int _seconds_since_start = seconds_since_start(seconds_start);
456 
457             Serial.println();
458             Serial.println();
459             Serial.println("STATE: WRITE SOCKET -- LASSI_CONTROLLER");
460             Serial.println();
461             // Serial.print("Seconds_since_start: ");
462             // Serial.println(_seconds_since_start);
463             if (production && !once){
464               state = STATE_PRODUCTION;
465             } else {
466               state = STATE_READ_SOCKET;
467             }
468 
469         }
470 
471 
472     }
473       break;
474 //=====================================================================WRITE
475       case STATE_PRODUCTION:
476 
477       //  while(!Serial.available()) Particle.process();
478         wrapper_production();
479         SdFatWrite(SdFatBuffer);
480         POST_SOCKET();
481   //    delay(800);
482         Serial.println();
483         state = STATE_READ_SOCKET;
484       break;
485 
486 //=====================================================================STATE CONNECT
487       case STATE_CONNECT:
```

```
488       //=================================================idle until key press
489         while(!Serial.available()) Particle.process();
490
491         Serial.println("connecting...");
492
493         if (client.connect(server, port)){
494
495           if (day<366){
496             GET();
497             state = STATE_READ;
498           } else {
499             state = STATE_DISCONNECT;
500           }
501
502         } else {
503           Serial.println("ERROR failed to connect");
504           client.stop();
505           delay(1000);
506         }
507       break;
508
509 //=================================================================READ
510       case STATE_READ:
511
512       if (client.available()){
513         _read();
514
515       } else if (!client.available()){
516         Serial.println();
517         Serial.println("STATE_READ: client unavailable");
518
519         // ================================================================STOP CLIENT GET
520         Serial.println();
521         client.flush();
522         Serial.println("disconnecting GET");
523         client.stop();
524         //=============================================================CALL LASSI
525         // wrapper();
526         // ================================================================STOP CLIENT POST
527         state = STATE_CONNECT;
528
529
530         //=========================================================REINITIALIZE READ VARIABLES
531         _read_reset();
532       }
533
534       break;
535
536       case TEST:
537         readPPFD();
538         read_amps();
539         writeAll(0x00);
540         Serial.println();
541         Serial.println();
542         Serial.println();
543         delay(1000);
544
545       break;
546
547
548       }
549
550
551 }
552 //
553 // void wrapper(){
554 //   //    const size_t bufferSize = JSON_ARRAY_SIZE(48) + 48*JSON_OBJECT_SIZE(1) + 700;
555 //   //    StaticJsonBuffer<bufferSize> jsonBuffer;
556 //   //    const int bufferSize = JSON_ARRAY_SIZE(48) + 48*JSON_OBJECT_SIZE(1);
557 //   //StaticJsonBuffer<bufferSize> jsonBuffer;
```

```
558 //   DynamicJsonBuffer jsonBuffer;
559 //   JsonArray& tree = jsonBuffer.parseArray(inChar);
560 //   //  tree.prettyPrintTo(Serial);
561 //   //    Serial.print("Array Size: ");
562 //   //    Serial.println(tree.size());
563 //
564 //   //=============================================================initialize transmit
565 //   //  JsonArray& packet =  jsonBuffer.createArray();
566 //   //=============================================================BEGIN LOOP OVER PACKET OF DATA
567 //   for (int i = 0;i<tree.size();i++){
568 //     //root[i].printTo(Serial);
569 //     // String t = tree[i]["T"].asString();
570 //     // char inputStr[11];
571 //     // t.toCharArray(inputStr,11);
572 //     // unsigned long _timevalue = atol(inputStr);
573 //     unsigned long _timevalue = tree[i]["T"];
574 //     Serial.print("Timevalue at read: ");
575 //     Serial.println(_timevalue);
576 //     Serial.println();
577 //
578 //
579 //     String l = tree[i]["L"].asString();
580 //     char input[7];
581 //     l.toCharArray(input,7);
582 //     float _PPFD = atof(input);
583 //     Serial.println();
584 //     Serial.print("PPFD value at read: ");
585 //     Serial.println(_PPFD);
586 //
587 //     LASSI(tree, timeClockOn, timeClockStart, timeClockFinish, i, _timevalue, _PPFD);
588 //
589 //   }
590 //
591 //   //=========================================================================END FOR LOOP
592 //   Serial.println();
593 //   tree.printTo(Serial);
594 //   Serial.println();
595 //   Serial.print("Array Size: ");
596 //   Serial.println(tree.size());
597 //
598 //   tree.printTo(buffer,sizeof(buffer));
599 //
600 //   if (tcp){
601 //     POST();
602 //     buff_reset();
603 //   }
604 //
605 //   //POST_SOCKET(tree);
606 //
607 //   //    Serial.print("DLI: ");
608 //   //    Serial.println(DLI_counter);
609 //
610 // }
611
612 //
613 // void wrapper_socket(){
614 //   //    const size_t bufferSize = JSON_ARRAY_SIZE(48) + 48*JSON_OBJECT_SIZE(1) + 700;
615 //   //    StaticJsonBuffer<bufferSize> jsonBuffer;
616 //   //    const int bufferSize = JSON_ARRAY_SIZE(48) + 48*JSON_OBJECT_SIZE(1);
617 //   //StaticJsonBuffer<bufferSize> jsonBuffer;
618 //   DynamicJsonBuffer jsonBuffer;
619 //   JsonArray& tree = jsonBuffer.parseArray(inChar);
620 //
621 //
622 //   //tree.prettyPrintTo(Serial);
623 //   Serial.println();
624 //
625 //   //=============================================================initialize transmit
626 //   //=============================================================BEGIN LOOP OVER PACKET OF DATA
627 //   for (int i = 0;i<tree.size();i++){
```

```
628 // //    Serial.println();
629 // /*
630 //     String l = tree[i]["L"].asString();
631 //     char input[7];
632 //     l.toCharArray(input,7);
633 //     float _PPFD = atof(input);
634 //     */
635 //
636 //     unsigned long _timevalue = tree[i]["T"];
637 //     Serial.print("Timevalue at read: ");
638 //     Serial.println(_timevalue);
639 //     _timevalue = ++_timevalue;
640 //     //
641 //     // float _PPFD = tree[i]["L"];
642 //     // Serial.print("new PPFD: ");
643 //     // Serial.println(_PPFD);
644 //     //
645 //
646 //
647 //     float _PPFD = simulate_daily_PPFD(_timevalue);
648 //     Serial.print("Simulate Daily PPFD: ");
649 //     Serial.println(_PPFD,6);
650 //
651 //     tree[i]["LL"] = 0.0;
652 //     tree[i]["E"] = 0.0;
653 //     tree[i]["D"] = 0.0;
654 //
655 //     if (once){
656 // //==================here DLI needs to be set to the last available value; from local storage
657 //       once = false;
658 //       float new_DLI = tree[i]["DLI"];
659 //       Serial.print("new DLI: ");
660 //       Serial.println(new_DLI);
661 //       DLI_counter = new_DLI;
662 //
663 //     }
664 //
665 //     LASSI(tree, timeClockOn, timeClockStart, timeClockFinish, i, _timevalue, _PPFD);
666 //
667 //   }
668 //
669 //   //=========================================================================END FOR LOOP
670 //   Serial.println();
671 //   Serial.print("POST: ");
672 //   tree.printTo(Serial);
673 //   Serial.println();
674 //   Serial.print("Array Size: ");
675 //   Serial.println(tree.size());
676 //
677 //   tree.printTo(buffer,sizeof(buffer));
678 //   tree[0].printTo(SdFatBuffer,sizeof(SdFatBuffer));
679 //
680 //   //    Serial.print("DLI: ");
681 //   //    Serial.println(DLI_counter);
682 //
683 // }
684
685 void initialize_production(JsonObject& nestedObject){
686   nestedObject["T"] = Time.now();
687   nestedObject["L"] = readPPFD();
688   nestedObject["LL"] = 0.0;
689   nestedObject["R"] = 0;
690   nestedObject["E"] = read_amps();
691   nestedObject["D"] = 0.0;
692   nestedObject["DLI"] = 0.0;
693
694   Serial.println();
695   Serial.println();
696
697 }
```

```
698
699 void wrapper_production(){
700   //    const size_t bufferSize = JSON_ARRAY_SIZE(48) + 48*JSON_OBJECT_SIZE(1) + 700;
701   //    StaticJsonBuffer<bufferSize> jsonBuffer;
702   //    const int bufferSize = JSON_ARRAY_SIZE(48) + 48*JSON_OBJECT_SIZE(1);
703   //StaticJsonBuffer<bufferSize> jsonBuffer;
704   DynamicJsonBuffer jsonBuffer;
705   JsonArray& tree = (once) ? jsonBuffer.parseArray(inChar) : jsonBuffer.createArray();
706
707
708   // //==================here DLI needs to be set to the last available value; from remote or local s
709   // Todo: make more robust by evaluating downtime and adjusting dli accordingly.
710   if (once) {
711     once = false;
712     tree.prettyPrintTo(Serial);
713     Serial.println();
714     float new_DLI = tree[0]["DLI"];
715     Serial.print("new DLI: ");
716     Serial.println(new_DLI,6);
717     DLI_counter = new_DLI;
718
719   } else {
720     JsonObject& nestedObject = tree.createNestedObject();
721     initialize_production(nestedObject);
722
723   }
724
725   // Serial.println("PRETTY PRINT TO SERIAL: ");
726   // tree.prettyPrintTo(Serial);
727   // Serial.println();
728
729   //=================================================================BEGIN LOOP OVER PACKET OF DATA
730   for (int i = 0;i<tree.size();i++){
731
732     unsigned long _timevalue = tree[i]["T"];
733     // Serial.print("Timevalue at read: ");
734     // Serial.println(_timevalue);
735     _timevalue = _timevalue;
736
737     float _PPFD = tree[i]["L"];
738     // Serial.print("new PPFD: ");
739     // Serial.println(_PPFD);
740
741     LASSI_production(tree, timeClockOn, timeClockStart, timeClockFinish, i, _timevalue, _PPFD);
742
743   }
744
745   //=============================================================================END FOR LOOP
746   Serial.println();
747   Serial.print("POST: ");
748   tree.prettyPrintTo(Serial);
749   Serial.println();
750   Serial.print("Array Size: ");
751   Serial.println(tree.size());
752
753   tree.printTo(buffer,sizeof(buffer));
754   tree[0].printTo(SdFatBuffer,sizeof(SdFatBuffer));
755
756 }
757 //=============================================================================LASSI
758 void LASSI_production(JsonArray& _packet, bool timeClockOn, int timeClockStart, int timeClockFinish,
759
760   int millisecondsOn = 1000;
761   byte controlByte = 0x00;
762
763   // if (binary){
764   //     PPFD_target = 700.0;
765   // }
766
767   if (algorithm_socket(timevalue, PPFD, timeClockOn, timeClockStart, timeClockFinish, PPFD_target)){
```

```
768   // if (false){
769     Serial.println("Lights off");
770     DLI_counter += milliseconds_PPFD(millisecondsOn,PPFD);
771     controlByte = 0x00;
772     writeAll(controlByte);
773 
774 
775 
776   } else {
777     Serial.println("Lights on");
778     DLI_counter += milliseconds_PPFD(millisecondsOn,PPFD);
779 
780     rule_flag = 0;
781 
782     Serial.print("DLI_counter: ");
783     Serial.println(DLI_counter,6);
784 
785     if (binary){
786       controlByte = 0x80;
787       writeAll(controlByte);
788 
789       Serial.print("controlByte: ");
790       Serial.println(controlByte);
791 
792 
793     } else {
794       controlByte = light_switch_new(PPFD, PPFD_target);
795       writeAll(controlByte);
796 
797       Serial.print("controlByte: ");
798       Serial.println(controlByte);
799     }
800 
801   };
802 
803 
804 //=========================================================transmission
805   Serial.println("Build Packet Production");
806   build_packet_production(_packet, i, timevalue, PPFD, rule_flag, controlByte);
807 
808 //=========================================================transmission end
809 
810 
811 // ========================================================= for loop end
812 }
813 
814 
815 //=========================================================================LASSI
816 // void LASSI(JsonArray& _packet, bool timeClockOn, int timeClockStart, int timeClockFinish, int i, u
817 //
818 //   int millisecondsOn = 1000;
819 //   float PPFD_650e = 200.0;
820 //   float PPFD_adjusted = 0.0;
821 //
822 //   if (tcp && tmy){
823 //     Serial.println("ran - 1 hour timestep");
824 //     millisecondsOn = 3600000;
825 //   } else if (tcp){
826 //     Serial.println("ran - half hour timestep");
827 //     millisecondsOn = 1800000;
828 //   }
829 //
830 //   if (algorithm_socket(timevalue, PPFD, timeClockOn, timeClockStart, timeClockFinish, PPFD_650e)){
831 //     //Serial.println(Time.hour(timevalue));
832 //     Serial.println("Lights off");
833 //     DLI_counter += milliseconds_PPFD(millisecondsOn,PPFD);
834 //     PPFD_adjusted = PPFD;
835 //
836 //   } else {
837 //
```

```
838 //     if (socket){
839 //       PPFD_650e = _packet[i]["LL"];
840 //     }
841 //
842 //     if (production){
843 //
844 //       PPFD_adjusted = PPFD;
845 //
846 //     } else {
847 //
848 //       PPFD_adjusted = PPFD + PPFD_650e;
849 //     }
850 //
851 //     DLI_counter += milliseconds_PPFD(millisecondsOn,PPFD_adjusted);
852 //
853 //
854 //     Serial.print("DLI_counter: ");
855 //     Serial.println(DLI_counter,6);
856 //
857 //     rule_flag = 0;
858 //
859 //     Serial.println("Lights on");
860 //
861 //   };
862 //
863 //
864 // //=========================================================transmission
865 //
866 // if (tcp){
867 //
868 //   Serial.println("Build Packet HTTP");
869 //
870 //   build_packet(_packet, i, timevalue, PPFD, PPFD_adjusted, rule_flag);
871 //
872 // } else if (socket) {
873 //
874 //   Serial.println("Build Packet Socket");
875 //   build_packet_socket(_packet, i, timevalue, PPFD, PPFD_adjusted, rule_flag);
876 //
877 // }
878 //
879 //
880 //
881 // //=========================================================transmission end
882 //
883 //
884 // // ========================================================= for loop end
885 // }
886 void build_packet_production(JsonArray& __packet, int _i, unsigned long timestamp, float PPFD, int _r
887
888   Serial.print("PPFD at build: ");
889   Serial.println(PPFD);
890   Serial.print("DLI at build: ");
891   Serial.println(DLI_counter);
892
893
894   __packet[_i]["T"] = timestamp;
895   __packet[_i]["L"] = PPFD*10000;
896 //  __packet[_i]["LL"] = PPFD_adjusted - PPFD;
897   __packet[_i]["R"] = rule_flag;
898   __packet[_i]["E"] = read_amps();
899   __packet[_i]["D"] = controlByte;
900   __packet[_i]["DLI"] = DLI_counter*10000000;
901
902 }
903 // void build_packet_socket(JsonArray& __packet, int _i, unsigned long timestamp, float PPFD, float P
904 //
905 //   Serial.print("PPFD at build: ");
906 //   Serial.println(PPFD);
907 //   Serial.print("DLI at build: ");
```

```
908 //   Serial.println(DLI_counter);
909 //
910 //
911 //   __packet[_i]["T"] = timestamp;
912 //   __packet[_i]["L"] = PPFD*10000;
913 //   __packet[_i]["LL"] = PPFD_adjusted - PPFD;
914 //   __packet[_i]["R"] = rule_flag;
915 // //  __packet[_i]["E"] = 0.0;
916 // //  __packet[_i]["D"] = 1.0;
917 //   __packet[_i]["DLI"] = DLI_counter*10000000;
918 //
919 // }
920 //
921 // void build_packet(JsonArray& __packet, int _i, unsigned long timestamp, float PPFD, float PPFD_adj
922 //
923 //   __packet[_i]["T"] = timestamp;
924 //   __packet[_i]["LL"] = PPFD_adjusted - PPFD;
925 //   __packet[_i]["R"] = rule_flag;
926 //
927 // }
928 float simulate_daily_PPFD(unsigned long timestamp){
929
930   float max = 100.0;
931   unsigned long xTerm = timestamp - 1483255381;
932 //  float sinTerm = 1.0/(86400.0)*3.14159;
933 //  float sinTerm = 1.0/(53096.0)*3.14159;
934   float sinTerm = 1.0/(86400.0/2)*3.14159;
935   float sin_curve = max * sinf(sinTerm*(float)xTerm);
936   float _final = constrain(sin_curve,0.0,max);
937   return _final;
938 }
939 //================================================================================LASSI RULES
940 bool algorithm_socket(unsigned long _timevalue, float _PPFD, bool timeClockOn, int timeClockStart, in
941
942   bool lightOff = true;
943
944   float latitude = 42.443961;
945   float longitude = -76.501881;
946
947   unsigned int _millisecondsInDay = 86400000;
948   unsigned int _millisecondsInHour = 3600000;
949   unsigned int _millisecondsInHalfHour = 1800000;
950   unsigned int _millisecondsInSecond = 1000;
951   unsigned int _offpeakStart = _millisecondsInDay - _millisecondsInHalfHour;
952
953   unsigned int _offpeakFinish = 7 * _millisecondsInHour;
954   unsigned int _darkperiod = 0;
955
956   int delaySummer = 12 * _millisecondsInHour; // May June July
957   int delayLateSummer = 9 * _millisecondsInHour; // August
958   int delaySpringFall = 7 * _millisecondsInHour; // March April September
959   int delayWinter = 5 * _millisecondsInHour; // January February October November December
960
961   int year = Time.year(_timevalue);
962   int month = Time.month(_timevalue);
963   int date = Time.day(_timevalue);
964   int hour = Time.hour(_timevalue);
965   int minute = Time.minute(_timevalue);
966   int second = Time.second(_timevalue);
967
968
969   Serial.print("year: ");
970   Serial.println(year);
971   Serial.print("month: ");
972   Serial.println(month);
973   Serial.print("date: ");
974   Serial.println(date);
975   Serial.print("hour: ");
976   Serial.println(hour);
977   Serial.print("minute: ");
```

```
978   Serial.println(minute);
979   Serial.print("second: ");
980   Serial.println(second);
981 
982   int current_time = hour*_millisecondsInHour + minute*60000 + second*_millisecondsInSecond;
983 
984   //A second preliminary calculation is made at 1:00 A.M. each day
985   //Calculate that day's sunrise and sunset hours
986   // based on the latitude and longitude of the greenhouse and the day of the year.
987 
988   Sunrise mySunrise(42,-76,-5);
989   mySunrise.Actual();
990 
991   int _SR = floor(mySunrise.Rise(month,date)*60*1000);
992   int _SS = ceil(mySunrise.Set(month,date)*60*1000);
993   int noon = mySunrise.Noon(month, date)*60*1000;
994   int _SR_hours = _SR/_millisecondsInHour;
995   int _SS_hours = (_SS/_millisecondsInHour)+1;
996   int current_time_hours = current_time/_millisecondsInHour;
997 
998   Serial.print("Current_time: ");
999   Serial.println(current_time);
1000   Serial.print("Sunrise ms: ");
1001   Serial.println(_SR);
1002   Serial.print("Sunset ms: ");
1003   Serial.println(_SS);
1004   Serial.print("current hour: ");
1005   Serial.println(current_time_hours);
1006   Serial.print("Sunrise Hours: ");
1007   Serial.println(_SR_hours);
1008   Serial.print("Sunset Hours: ");
1009   Serial.println(_SS_hours);
1010 
1011 
1012 //================================================================DLI Reset
1013 
1014 
1015 if (tcp){
1016   Serial.println("----------------------------------Ran STATE_READ in LASSI");
1017 
1018   if (current_time_hours + minute == _SR_hours){
1019     DLI_counter = 0.0;
1020     Serial.println("Ran DLI_counter Reset -- STATE_READ");
1021     current_time = current_time_hours*_millisecondsInHour;
1022   }
1023 
1024   _SR = _SR_hours*_millisecondsInHour;
1025 
1026 } else {
1027 
1028   if (current_time == _SR){
1029     DLI_counter = 0.0;
1030     Serial.println("Ran DLI_counter Reset");
1031   }
1032 
1033 }
1034 //============================================================ just for testing; delete during product
1035 if (socket){
1036   if (current_time < _SR || current_time > _SS){
1037     _PPFD = 0.0;
1038   }
1039 
1040 }
1041 
1042 
1043   float idealPPFDtoTime = idealPPFDSunriseToTime(_SR,_SS,current_time,DLItarget);
1044   float idealPPFDtoNoon = idealPPFDSunriseToTime(_SR,_SS,noon,DLItarget);
1045 
1046   float totalDLIdeficit = constrain((DLItarget - DLI_counter), 0.00, DLItarget);
1047   float currentDLIdeficit = constrain((idealPPFDtoTime - DLI_counter), 0.00, DLItarget);
```

```
1048
1049
1050 // this calculation is wrong and should be fixed, specifcially when current_time < sunrise
1051 //  float safetyCheck = lightIntegral_650e(_millisecondsInDay - current_time + _SR);
1052
1053
1054   //  Serial.print("safety Check: ");
1055   //  Serial.println(safetyCheck);
1056
1057   Serial.print("idealPPFDtoTime: ");
1058   Serial.println(idealPPFDtoTime);
1059   Serial.print("idealPPFDtoNoon: ");
1060   Serial.println(idealPPFDtoNoon);
1061
1062   Serial.print("DLI target: ");
1063   Serial.println(DLI_counter+totalDLIdeficit);
1064   Serial.print("DLI_counter: ");
1065   Serial.println(DLI_counter,6);
1066   Serial.print("total DLI deficit: ");
1067   Serial.println(totalDLIdeficit);
1068   Serial.print("current DLI deficit: ");
1069   Serial.println(currentDLIdeficit);
1070 /*
1071   Serial.print("_offpeakStart: ");
1072   Serial.println(_offpeakStart);
1073   Serial.print("_offpeakFinish: ");
1074   Serial.println(_offpeakFinish);
1075   Serial.print("_millisecondsInDay: ");
1076   Serial.println(_millisecondsInDay);
1077   Serial.print("_darkperiod: ");
1078   Serial.println(_darkperiod);
1079 */
1080   //A preliminary calculation is made only at the first hour of the weather data set:
1081   //The integrated supplemental PPFD achievable by operating lights during the entire off-peak perio
1082   //(less a possible dark period for photoperiod control).
1083   // This assumes time-of-day electricity rates do not change during the year.
1084
1085   unsigned int __milliseconds_offpeak = _milliseconds_offpeak(_offpeakStart,_offpeakFinish,_millisec
1086   float supplemental_offpeak_potential = lightIntegral_650e(__milliseconds_offpeak, PPFD_650e);
1087   float scaledOffPeakPotential = scale_offpeak_potential(_SR, _SS, current_time)/100.00*supplemental
1088
1089
1090
1091
1092   //A third preliminary calculation is made for each hour of the weather data set
1093   //The total (potential) PPFD that could be accumulated using only supplemental lighting
1094   //if lamps were to be on starting at the beginning of the next hour
1095   //and remain on until the following sunrise hour or end of the off-peak period
1096   //(and possibly de-activated for a dark period for photoperiod purposes),
1097   // whichever comes first.
1098
1099   unsigned int milliseconds_until_offpeak = __supplemental_PPFD_potential(current_time, _SR, _darkpe
1100   float supplemental_PPFD_potential = lightIntegral_650e(milliseconds_until_offpeak, PPFD_650e);
1101 /*
1102   Serial.print("__milliseconds_offpeak: ");
1103   Serial.println(__milliseconds_offpeak);
1104   Serial.print("Supplemental off-peak potential: ");
1105   Serial.println(supplemental_offpeak_potential);
1106   Serial.print("Scaled off peak potential: ");
1107   Serial.println(scaledOffPeakPotential);
1108   Serial.print("milliseconds_until_offpeak: ");
1109   Serial.println(milliseconds_until_offpeak);
1110   Serial.print("Supplemental PPFD potential: ");
1111   Serial.println(supplemental_PPFD_potential);
1112 */
1113
1114 // RULE 1: If time clock control is included and the current hour is during the period when lamps sh
1115 //    control is activated and the lamps are turned off.
1116 //    Time clock control was inactive for everything reported herein.
1117 //    Lettuce was the crop of interest and lettuce needs no dark period to thrive.*/
```

```
1118   if (timeClockOn && (current_time < timeClockStart || current_time > timeClockFinish))
1119   {
1120     Serial.println("ran rule 1");
1121     rule_flag = 1;
1122     lightOff = true;
1123     return lightOff;
1124   }
1125   else if (timeClockOn && (current_time > timeClockStart || current_time < timeClockFinish))
1126   {
1127     Serial.println("ran rule 1");
1128     rule_flag = 0;
1129     lightOff = false;
1130     return lightOff;
1131   }
1132
1133   //  RULE 2a: For months of greatest solar irradiation, keep lamps off between sunrise and H1 hours
1134   // However, if the daily accumulated PPFD is not equal to at least one-quarter of the daily target
1135   // permit lights to remain on regardless of the value of H1.
1136   else if ((month == 5 || month == 6 || month == 7) && (current_time >= _SR && current_time < ( _SR
1137   {
1138
1139     if ((current_time >= noon) && (DLI_counter < (0.25 * DLItarget )))
1140     {
1141       Serial.println("ran rule 2a");
1142       rule_flag = 0;
1143       lightOff = false;
1144       return lightOff;
1145     }
1146     else
1147     {
1148       Serial.println("ran rule 2a");
1149       rule_flag = 2;
1150       lightOff = true;
1151       return lightOff;
1152     }
1153   }
1154
1155 // RULE 2b:
1156 //  For late summer (when days are still sunny, but solar intensity has lessened),
1157 //  keep lamps off between sunrise and H2 hours after sunrise.
1158 //  However, if the daily accumulated PPFD is not equal to at least one-quarter of the daily target
1159 //  permit lights to remain on regardless of the value of H2
1160   else if ((month == 8) && (current_time > _SR && current_time < ( _SR + delayLateSummer)))
1161   {
1162
1163     if ((current_time >= noon) && (DLI_counter < (0.25 * DLItarget)))
1164     {
1165       Serial.println("ran rule 2b");
1166       rule_flag = 0;
1167       lightOff = false;
1168       return lightOff;
1169     }
1170     else
1171     {
1172       Serial.println("ran rule 2b");
1173       rule_flag = 3;
1174       lightOff = true;
1175       return lightOff;
1176     }
1177
1178   }
1179
1180 // RULE 2c: For spring and autumn months, keep lamps off between sunrise and H3 hours after sunrise.
1181   else if ((month == 3 || month == 4 || month == 9) && ( current_time >= _SR && current_time < ( _SR
1182   {
1183     Serial.println("ran rule 2c");
1184     rule_flag = 4;
1185     lightOff = true;
1186     return lightOff;
1187   }
```

```
1188
1189 //RULE 2D:
1190 //  For the rest of the months of the year,
1191 //  keep lamps off between sunrise and H4 hours after sunrise.
1192   else if ((month == 1 || month == 2 || month == 10 || month == 11 || month == 12) && ( current_time
1193   {
1194     Serial.println("ran rule 2d");
1195     rule_flag = 5;
1196     lightOff = true;
1197     return lightOff;
1198   }
1199
1200 //RULE 3: If solar PPFD accumulated to this hour meets or exceeds the accumulation target (eq. 6) fo
1201 //  turn the lights off. Justification: To this hour, there is no PPFD integral deficit.
1202   else if (DLI_counter >= idealPPFDtoTime)
1203   {
1204     Serial.println("ran rule 3");
1205     rule_flag = 6;
1206     lightOff = true;
1207     return lightOff;
1208   }
1209
1210 //RULE 4: If: (a) the hour is during the time of year with more sunlight and between sunrise and sun
1211 //  (b) the PPFD left to be accumulated can be achieved by delaying supplemental lighting until the
1212 //  and the PPFD deficit to this point could be made up by a scaled portion of the off-peak PPFD pot
1213 //  turn off the lights.
1214   else if ((month == 5 || month == 6 || month == 7 || month == 8) && (current_time >= _SR && current
1215   {
1216     if ((totalDLIdeficit < supplemental_PPFD_potential) && (currentDLIdeficit < scaledOffPeakPotenti
1217     {
1218       Serial.println("ran rule 4");
1219       rule_flag = 7;
1220       lightOff = true;
1221       return lightOff;
1222     }
1223   }
1224
1225 //RULE 5: Turn off the lights if the hour is between sunrise and sunset
1226 //  and the PPFD left to be accumulated could be accumulated by turning on the lights at the next ho
1227 //  even if the solar PPFD drops immediately to insignificance and remains there for the rest of the
1228   else if ((current_time > _SR && current_time < _SS) && totalDLIdeficit < supplemental_PPFD_potenti
1229   {
1230     Serial.println("ran rule 5");
1231     rule_flag = 8;
1232     lightOff = true;
1233     return lightOff;
1234   }
1235
1236 //RULE 6: If: (a) the hour is at sunset or between sunset and an hour before the start of off-peak e
1237 //  (b) the accumulated PPFD deficiency to this hour could be achieved during off-peak hours alone,
1238   else if ((current_time >= _SS && current_time < (_offpeakStart - _millisecondsInSecond)) && totalD
1239   {
1240     Serial.println("ran rule 6");
1241     rule_flag = 9;
1242     lightOff = true;
1243     return lightOff;
1244   }
1245
1246 //RULE 7: If the hour is before off-peak electric rates start,
1247 //  but any remaining PPFD to be added by supplemental lighting will be achieved before the off-peak
1248 //  turn off the lights.
1249   else if (current_time < _offpeakStart && current_time > _offpeakFinish && totalDLIdeficit < supple
1250   {
1251     Serial.println("ran rule 7");
1252     rule_flag = 10;
1253     lightOff = true;
1254     return lightOff;
1255   }
1256 //Be sure lights are not turned off
1257 //if the hour is during the dark part of the year
```

```
1258 // and there remains more integrated PPFD to be added than can be met by the lamps alone,
1259 // operating from the next hour until the following sunrise
1260   if ((month == 10 || month == 11 || month == 12 || month == 1 || month == 2) && totalDLIdeficit > s
1261   {
1262     Serial.println("ran rule 8");
1263     rule_flag = 11;
1264     lightOff = false;
1265     return lightOff;
1266   }
1267
1268 }
1269
1270 //================================================SUPPORT FUNCTIONS
1271
1272 unsigned int seconds_since_start(int _seconds_start) {
1273   unsigned int seconds_current = Time.local();
1274   unsigned int time_since_start = seconds_current - _seconds_start;
1275   return time_since_start;
1276 }
1277
1278 float scale_offpeak_potential (int _SR, int _SS, int _timeT){
1279
1280   float scale_factor= constrain(((float)map(_timeT, _SR, _SS, 0, 100)), 0, 100);
1281
1282   return scale_factor;
1283 }
1284
1285 float idealPPFDSunriseToTime(int _SR, int _SS, int _timeT, float _PPFDtarget){
1286   // Account for hours after sunset and before sunrise
1287   // during which the ideal DLI to the hour is a function of the previous day.
1288   if (_timeT >= _SS || _timeT < _SR)
1289   {
1290     _timeT = _SS;
1291   }
1292
1293   float PPFDintegral = .5 * _PPFDtarget * (1-cos(3.14159*(((float)_timeT-(float)_SR)/((float)_SS-(fl
1294
1295   return PPFDintegral;
1296 }
1297
1298 unsigned int __supplemental_PPFD_potential(int current_time, int _SR, unsigned int darkPeriod, unsig
1299
1300   unsigned int hoursTillNextSunrise = 0;
1301   unsigned int millisecondsInDay = 86400000;
1302   unsigned int millisecondsInHour = 3600000;
1303
1304   if ((current_time+1000) > millisecondsInDay){
1305     current_time += 1000;
1306   }else{
1307     current_time = 1000;
1308   }
1309
1310   if (_SR < offpeakFinish){
1311
1312     if (current_time < _SR ) {
1313       hoursTillNextSunrise = (_SR - current_time) - darkPeriod;
1314     } else {
1315       hoursTillNextSunrise = (millisecondsInDay - current_time) + _SR - darkPeriod;
1316     }
1317
1318   }else{
1319
1320     if (current_time < offpeakFinish ) {
1321       hoursTillNextSunrise = (offpeakFinish - current_time) - darkPeriod;
1322     } else {
1323       hoursTillNextSunrise = (millisecondsInDay - current_time) + offpeakFinish - darkPeriod;
1324     }
1325
1326   }
1327
```

```
1328   return hoursTillNextSunrise;
1329
1330 }
1331
1332 float lightIntegral_650e (int millisecondsOn, float PPFD_650e){
1333   float PPFDoutput = PPFD_650e;
1334   int secondsInHour = 3600;
1335   float avogadros = 1000000.0;
1336   float millisecondsInHour = 3600000.0;
1337
1338   return (PPFDoutput*secondsInHour/avogadros)*(millisecondsOn/millisecondsInHour);
1339 }
1340
1341 float milliseconds_PPFD(int millisecondsOn, float PPFD){
1342   int secondsInHour = 3600;
1343   float avogadros = 1000000.0;
1344   float millisecondsInHour = 3600000.0;
1345
1346   return (PPFD*secondsInHour/avogadros)*((float)millisecondsOn/(float)millisecondsInHour);
1347 }
1348
1349 unsigned int _milliseconds_offpeak(int __offpeakStart,int __offpeakFinish,int __millisecondsInDay,in
1350   unsigned int milliseconds_offpeak;
1351
1352   if (__offpeakFinish > __offpeakStart){
1353     milliseconds_offpeak = (__offpeakFinish - __offpeakStart) - __darkperiod;
1354     } else {
1355       milliseconds_offpeak = ((__millisecondsInDay - __offpeakStart) + __offpeakFinish) - __darkperi
1356     }
1357     return milliseconds_offpeak;
1358 }
1359
1360 int day365(unsigned long _timevalue){
1361   int year = Time.year(_timevalue);
1362   int month = Time.month(_timevalue);
1363   int day = Time.day(_timevalue);
1364
1365   int n1 = (275*month/9);
1366   int n2 = ((month + 9)/12);
1367   int n3 = (1+(year-4*(year/4+2)/3));
1368   int n = n1 - (n2*n3) + day - 30;
1369   //Serial.println(n);
1370   return n;
1371 }
1372
1373
1374
1375 //================================================PWM SUPPORT FUNCTIONS
1376 float licor_sensor_PPFD (){
1377   int sum = 0;
1378   int sensor_input = 0;
1379
1380   for (int i = 0; i < numSamples; i++){
1381     sensor_input = analogRead(licor_sensor_pin);
1382     delay(1);
1383     sum += sensor_input;
1384   }
1385
1386   float reading = (float)sum/(float)numSamples;
1387
1388   reading = float_map(reading, 0.0, 4095.0, 0.0, 4095.0);
1389   reading = constrain(reading, 0.0, 4095.0);
1390   Serial.print("licor analog read: ");
1391   Serial.println(reading);
1392
1393   float volts = (reading*3.3) / 4095.0;
1394   Serial.print("licor volts: ");
1395   Serial.println(volts,6);
1396   return volts;
1397 }
```

```
1398
1399 float volts_to_micromoles(float volts){
1400   //==================================CONVERT VOLTS TO MICROMOLES
1401   float calibration_factor = 0.0;
1402
1403   if (binary){
1404
1405     calibration_factor = (702.2)/(547.06);
1406
1407   } else {
1408
1409     //calibration 1
1410     calibration_factor = (540.0 / 496.83);
1411
1412   }
1413
1414   float umols = constrain(float_map((volts*calibration_factor), 0.0, 3.3, 0.0, 2500.0),0.0,2500.0);
1415   Serial.print("licor umols: ");
1416   Serial.println(umols);
1417
1418   return umols;
1419 }
1420
1421 float readPPFD() {
1422
1423   float _licor_volts = licor_sensor_PPFD();
1424
1425   float _filtered_signal = low_pass_filter(filtered_signal_previous, _licor_volts);
1426   Serial.print("Filtered Signal: ");
1427   Serial.println(_filtered_signal,6);
1428
1429   float PPFD_actual = volts_to_micromoles(_filtered_signal);
1430
1431   return PPFD_actual;
1432 }
1433
1434 void PWM_control_calc(float PPFD_actual) {
1435
1436   float photoperiod = 24;
1437
1438   float PPFD_target = _PPFD_target(photoperiod, DLItarget);
1439
1440
1441   light_switch_new(PPFD_actual, PPFD_target);
1442 //  return PWM_control;
1443
1444 }
1445
1446 float _PPFD_target (float _photoperiod, float _DLI_target){
1447   int avogadros = 1000000;
1448   int seconds = _photoperiod*60*60;
1449   float PPFD_target = _DLI_target/(float)seconds*(float)avogadros;
1450   Serial.print("PPFD_target: ");
1451   Serial.println(PPFD_target);
1452
1453   return PPFD_target;
1454 }
1455
1456
1457 byte light_switch_new(float umols_actual, float umols_target){
1458     Serial.println("PPFD_target: ");
1459     Serial.println(umols_target);
1460       // float difference = constrain((umols_target - umols_actual),0.0,umols_target);
1461     float difference = (umols_target - umols_actual);
1462     Serial.print("difference: ");
1463     Serial.println(difference);
1464
1465     byte controlByte = readAll();
1466     float scale_factor = difference/umols_target;
1467     Serial.print("scale_factor: ");
```

```
1468     Serial.println(scale_factor);
1469
1470     controlByte += (128*scale_factor);
1471     controlByte = constrain(controlByte, 0, 128);
1472
1473     Serial.print("controlByte: ");
1474     Serial.println(controlByte);
1475
1476     return controlByte;
1477 }
1478
1479 // int light_switch(float umols_actual, float umols_target, float PPFD_full_power){
1480 //   float difference = constrain((umols_target - umols_actual),0.0,umols_target);
1481 //   Serial.print("difference: ");
1482 //   Serial.println(difference);
1483 //   float _PWM_control = float_map(difference,0.0,PPFD_full_power,15.0,90.0);
1484 //
1485 //
1486 //
1487 //   Serial.print("PWM_control: ");
1488 //   Serial.println(_PWM_control);
1489 //   return _PWM_control;
1490 // }
1491
1492 float float_map(float x, float in_min, float in_max, float out_min, float out_max){
1493   return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
1494 }
1495
1496 float low_pass_filter(float _filtered_signal_previous, float _raw_signal){
1497   float alpha = 0.4;
1498   float filtered_signal_current = _filtered_signal_previous * (1-alpha) + (_raw_signal * alpha);
1499   filtered_signal_previous = filtered_signal_current;
1500   return filtered_signal_current;
1501
1502 }
1503
1504 void SdFatWrite(char toWrite[]){
1505   //
1506   // if (!Serial.available()){
1507   //   Serial.println("Type any character to start");
1508   //   while(!Serial.available()) Particle.process();
1509   // }
1510   //
1511   // open the file for write at end like the "Native SD library"
1512   if (!myFile.open("test2.txt", O_RDWR | O_CREAT | O_AT_END)) {
1513   //  sd.errorHalt("opening test.txt for write failed");
1514       Serial.println();
1515       Serial.println("SDFAT ERROR: open file failed...");
1516   }
1517
1518   // if the file opened okay, write to it:
1519   Serial.print("Writing to test.txt...");
1520   myFile.println(String(toWrite)+",");
1521 //  myFile.printf("fileSize: %d\n", myFile.fileSize());
1522
1523   // close the file:
1524   myFile.close();
1525   SdFat_reset();
1526   Serial.println(" done.");
1527
1528 }
1529
1530 void SdFatRead(){
1531
1532   // re-open the file for reading:
1533   if (!myFile.open("test.txt", O_READ)) {
1534   //  sd.errorHalt("opening test.txt for read failed");
1535   Serial.println("opening test.txt for read failed");
1536   }
1537   Serial.println();
```

```
  Serial.println("FILE CONTENTS: ");

  // read from the file until there's nothing else in it:
  int data;
  while ((data = myFile.read()) >= 0) {
    Serial.write(data);
  }
  // close the file:
  myFile.close();

}


float read_amps(){

  float vin = 4.59;
  float Vref = vin /2.0;
  float mvPerAmp = 185.0;
  float cal_const = 0.0;

  if (binary) {
     cal_const = (5.35)/(4.44);
  } else {
     cal_const = (5.35)/(4.37);
  }

  float sensorValue[numSamples];
  float sum = 0;
  double _sum = 0;

  for (int i = 0; i < numSamples; i++){

    float volts = (((float)analogRead(energy_pin)/4095.0*3.3))-Vref ;
    delay(1);

    float result = volts;
    sensorValue[i] = result;
    sum += result;

  }
  //
  //
  float mean = sum / (float)numSamples;

  for (int i = 0; i < numSamples; i++){
    _sum += pow((sensorValue[i] - mean),2);
    sensorValue[i] = '\0';
  }

  double sigma = sqrt((_sum/numSamples));
  Serial.print("Sigma: ");
  Serial.println(sigma,6);

  float millivolts = sigma*1000.0;
  Serial.print("Millivolts: ");
  Serial.println(millivolts,6);

  float amps = millivolts / mvPerAmp * cal_const;

  Serial.print("Amps: ");
  Serial.println(amps,6);

  return amps;
}

void low_med_high(byte* values){

    int c = 0;

    if (Serial.available()>0){
```

```
      c = Serial.read();


    }

    switch(c){

      case 49:
        Serial.println("Ran 1");
        writeAll(values[0]);
        delay(1000);

      break;

      case 50:
        Serial.println("Ran 2");
        writeAll(values[1]);
        delay(1000);

      break;

      case 51:
        Serial.println("Ran 3");
        writeAll(values[2]);
        delay(1000);

      break;



    }

}

void writeAll(byte valueWrite){
  write(address,wiperRedW,valueWrite);
  write(address,wiperBlueW,valueWrite);
  write(address,wiperWhiteW,valueWrite);

}

void write(byte address, byte device_address, byte value){
  int length = 2;
  byte transmission[length] = {device_address, value};
  byte messages[3];
  Wire.beginTransmission(address);
  messages[1] = Wire.write(transmission,length);
  messages[2] = Wire.endTransmission(true);
  messages[3] = 0;
  print_messages(messages);
}

void cycle(){
  int counter = readAll();
  Serial.print("ReadValue: ");
  Serial.println(counter, DEC);
  if (flip){
    incrementDecrementAll(true);
    if (counter >= 128){
      flip = false;
    }
  } else {
    incrementDecrementAll(false);
    if (counter <= 0){
      flip = true;
    }
  }
}

void incrementDecrementAll(bool increment){
```

```
1678   if (increment){
1679     incrementDecrement(address,wiperRedI);
1680     incrementDecrement(address,wiperBlueI);
1681     incrementDecrement(address,wiperWhiteI);
1682   }else{
1683     incrementDecrement(address,wiperRedD);
1684     incrementDecrement(address,wiperBlueD);
1685     incrementDecrement(address,wiperWhiteD);
1686   }
1687 }
1688
1689 void incrementDecrement(byte address, byte device_address){
1690
1691   byte messages[3];
1692   Wire.beginTransmission(address);
1693   messages[1] = Wire.write(device_address);
1694   messages[2] = Wire.endTransmission(true);
1695   messages[3] = 0;
1696   print_messages(messages);
1697 }
1698
1699 int readAll(){
1700   int valueRED = read(address, wiperRedR);
1701   int valueBLUE = read(address, wiperBlueR);
1702   int valueWHITE = read(address, wiperWhiteR);
1703   // Serial.print("ValueRED: ");
1704   // Serial.println(valueRED, DEC);
1705   // Serial.print("ValueBLUE: ");
1706   // Serial.println(valueBLUE, DEC);
1707   // Serial.print("ValueWHITE: ");
1708   // Serial.println(valueWHITE, DEC);
1709   return valueRED;
1710 }
1711
1712 int read(byte address, byte device_address){
1713   byte messages[3];
1714   int numBytesRequested = 2;
1715   Wire.beginTransmission(address);
1716   messages[1] = Wire.write(device_address);
1717   messages[2] = Wire.endTransmission(false);
1718   messages[3] = Wire.requestFrom(address, numBytesRequested,true);    // request 6 bytes from slave
1719 //  print_messages(messages);
1720   while(Wire.available()){   // slave may send less than requested
1721       char c = Wire.read();    // receive a byte as character
1722       if (Wire.available() < numBytesRequested-1){
1723         // Serial.print(c,BIN);
1724         // Serial.println();
1725         // Serial.print("Read value: ");
1726         // Serial.println(c, DEC);
1727         // Serial.println();
1728         return c;
1729       }
1730   }
1731   delay(10);
1732 }
1733
1734
1735
1736 void print_messages(byte* messages){
1737   byte written, received, error;
1738   written = messages[1];
1739   error = messages[2];
1740   received = messages[3];
1741
1742   if(error != 0){
1743
1744     Serial.print("Error: ");
1745     Serial.println(error);
1746
1747
```

```
1748   } else if (written != 0) {
1749
1750     Serial.print("Wrote ");
1751     Serial.print(written);
1752     Serial.println(" Bytes!");
1753
1754   }
1755
1756   if (received != 0) {
1757
1758     Serial.print("Received ");
1759     Serial.print(received);
1760     Serial.println(" Bytes!");
1761
1762   }
1763
1764   Serial.println();
1765
1766
1767
1768 }
1769
```