Building a framework to estimate photosynthetic yield by CO_2 consumption

Searching for the action spectrum with a 3-channel LED array

Bjørnar Prytz

Thesis submitted for the degree of
Master in Network and system administration
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2018

Building a framework to estimate photosynthetic yield by CO_2 consumption

Searching for the action spectrum with a 3-channel LED array

Bjørnar Prytz

Abstract

This thesis contains the description of the design and implementation of a basic framework for research on how plants behave in controlled environments. The framework is capable of controlling lighting with a pair of 3-channel LED arrays, and of estimating photosynthetic action of incubated plants. These features are used in an experiment to rate a selection of light settings by how well they stimulate photosynthetic action.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

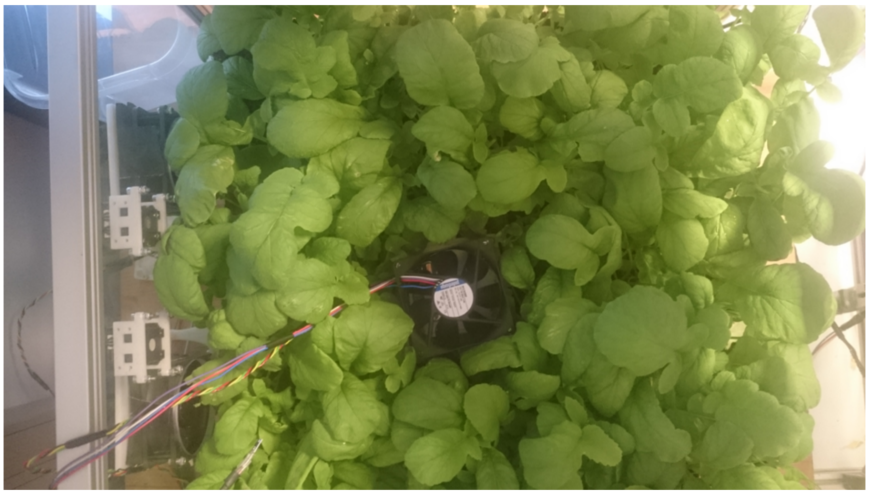


Figure 1.1: The inside of the enclosure, with the internal sensor (/w fan attached) in the middle.

1.1 Goal and Motivation

With humanity starting to send cars into space[15], and the public imagination already dreaming of the red planet, the question of how to feed interplanetary travellers is unavoidable.

Back on Earth, the domain of indoor horticulture intersects with home gardening, and even larger scale production. The efficient use of space in vertical farming has the potential to alter food distribution networks, especially in dense urban centers, and dreams of such a fundamental change in an essential industry can send the imagination of an idealist spinning.

Reducing energy consumption is another way to improve production efficiency, and grow lights naturally consume a large portion of the energy in these systems.

The goal of this project is to provide a basic framework for further research on how plants behave in controlled environments. Additionaly it examines the possibility for that framework to estimate the photosynthetic action of incubated plants through the measurement of only CO_2.

These estimations are used to rate different settings on a pair of adjustable 3-channel LED arrays. The framework uses an energy allowance to keep the consumption equal between all settings, so that the photosynthetic rate becomes a measurement of energy efficiency.

1.2 Project Description

This thesis describes the design and implementation of a framework with which to control and measure an enclosure (figure 1.1) with plants in. The plexiglass enclosure has dampers, fans and a pair of LED arrays for climate control, and sensors for monitoring. All of these components are controlled by an Arduino, which runs part of the framework code.

The rest of the software is on a Raspberry Pi (RPi), which controls the Aruduino through a protocol that was written for this framework. The RPi takes care of higher level routines (e.g. experiment procedures, day-night cycles), and is designed to be expanded and used in other applications.

It is already established that plants have different reactions to the various spetra of light, and those reactions may depend on myriad factors (eg. plant species, temperature, time of day). The thesis asks whether it is possible to estimate those reactions by only measuring the CO_2 concentrations within the enclosure. The method to be tested is outlined in figure 1.2.

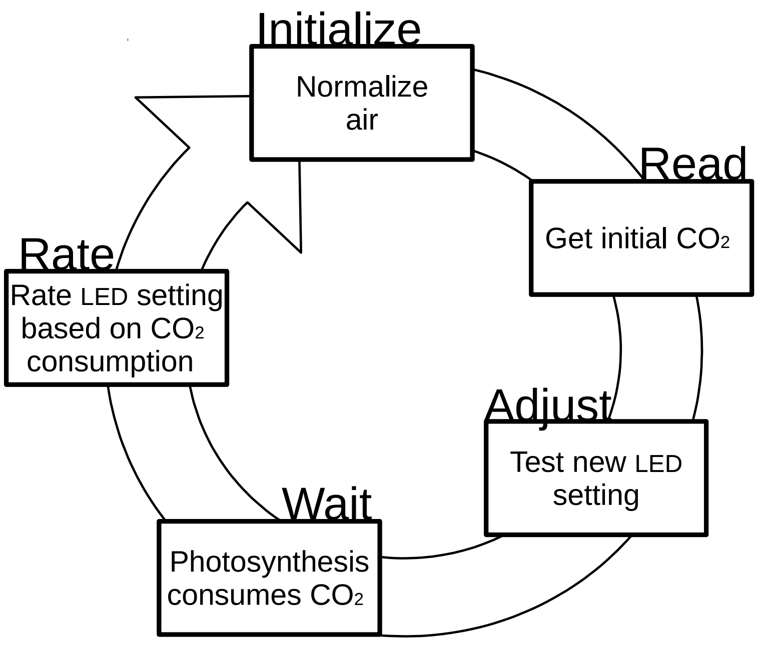


Figure 1.2: The main loop of a framework designed to adapt its behaviour to the response of a plant

1.3 Hypothesis

The following main and sub hypotheses were arrived at:

1. What is required to be able to tune a 3-channel LED array to optimize photosynthetic action within a given energy allowance?
    (a) Is it possible to estimate photosynthetic action reliably by only measuring CO_2 concentration over time?
    (b) How long does an interval need to be in order to measure the fitness of an LED setting?
    (c) What sources of disturbance are there in the experiments?

1.4 Outline

The following thesis will describe the background that needed to be understood in order to carry out the theoretical and practical parts of this project.

Following that is a chapter describing the design and implementation of a framework with hardware and software elements. The hardware is designed around an enclosure for growing plants, and the components that control and measure that enclosure. The software portion delves into the control routines and interfaces that are involved, along with data flow from sensor data to visual presentation.

The next chapter describes the methodology of the experiments that were carried out in order to answer the hypotheses laid out above.

Then the results of the experiments are presented, and in the concluding chapter, they are discussed, and the hypotheses revisited. Finally, some potential avenues for future work are mentioned.

Chapter 2

Technical Background

2.1 The Potential of Artificial Lighting

Massa et al. carried out research into the effects of different narrow wavebands of light on plant physiology[21]. In the paper they point to the various and complex effects of light on plant physiology: Flowering[12] and internode elongation[23] can be stimulated by far-red light(710-850nm); blue light affects a different range of properties[25], [28], [33].

Heo et al. found similar effects when exploring LED as an alternative to fluorescent lamps as grow lights. Both studies mention space-based horticulture[17].

2.2 Plant Physiology

The biochemical processes of plant physiology are as intricate as they are ancient. A complete understanding of the finesse of photosynthesis is outside the scope of this thesis, rather the goal of this section is to map out some potentially important factors to consider in order to conduct meaningful experiments.

2.2.1 Photosynthesis

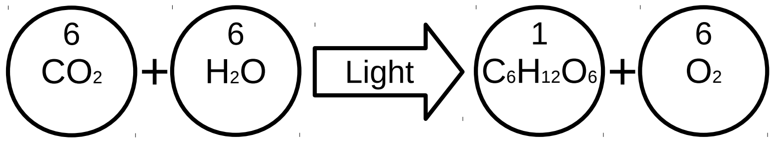


Figure 2.1: Illustration of the reactants and products of photosynthesis. It takes 6 CO_2 molecules and 6 H_2O molecules to synthesize one molecule of carbohydrates and 6 O_2 molecules

Photosynthesis(table 2.1) can be divided into two main phases: *Light-dependent reactions*, and *light-independent reactions* (Calvin cycle[7]). Each phase provides necessary chemicals for the other phase, and they do occur simultaneously, despite their names suggesting otherwise. These chemical reactions are quite complex, but they will be simplified profusely to get across the main points relevant to this thesis.

Light-Dependent Reactions

The goal of the light-dependent reactions[5] is to store the energy that erupts when light is absorbed by the leaf. These energy stores (called ATP) are synthesized from (among other compounds) the hydrogen in water, and oxygen (O_2) is a byproduct of this reaction. The ATP molecules are later used by the Calvin cycle, but its purpose beyond that is not within our scope here.

The Calvin Cycle

Also aptly called the light-independent reactions[7], part of the Calvin cycle's role in photosynthesis is to fixate (bind to a new molecule) carbon into an intermediary compound which eventually becomes glucose. The carbon is harvested from CO_2 in the air. By measuring the rate of this carbon fixation, it is possible to measure the rate of the photosynthetic cycle.

Photosystems and the Emerson Enhancement Effect

There are two structures within plant cells which have the task of turning light into energy and transporting that energy to the various reaction centers[24]. These structures are named photosystem II (PS-II) and photosystem I (PS-I). This pair is key in an electron transport chain that powers the light-dependent ractions2.2.

PS-II is able to tear electrons from water molecules (H_2O) and energize them with the energy from captured photons (light). Energized electrons are sent along a transport chain to PS-I, where they are re-energized and forwarded to the next recipient. The energy that is taken from the electron as it hops through the chain is used for many different purposes that will not be detailed here.

Each of the photosystems contains pigments that are tuned to a different wavelength of light, and their names leave no doubt as to their attunemnet. P680 is the pigment that resides in PS-II, and P700 is the one in PS-II. Whenever both of these wavelengths (680 nm and 700 nm) of light are present at the same time a synergy effect(Emerson Enhancement[14]) occurs that is greater than the sum of the effect of stimulating PS-II and PS-I separately.

2.2.2 Photosynthetically Active Radiation

The light that is utilized by a plant for photosynthetic action is called photosynthetically active radiation (PAR)[22]. The range can vary depending

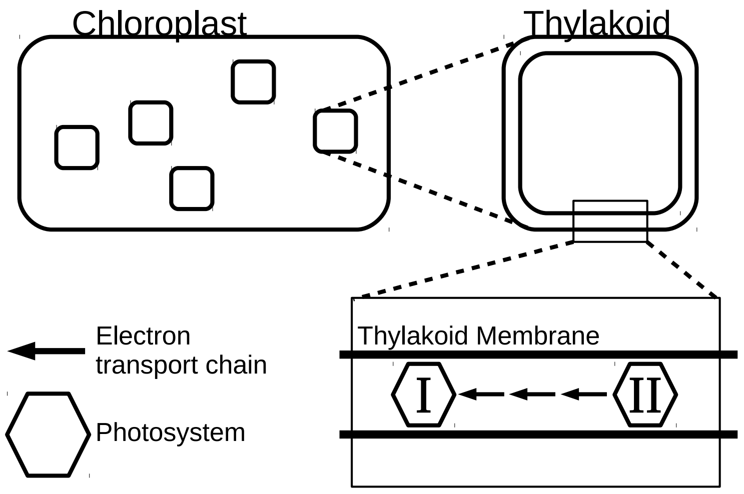


Figure 2.2: The photosystems[24] are complex structures within the thylakoid membrane specialized in absorbing certain wavelengths of light. The reaction center of photosystem II (PS-II) is named P680 because 680 nm light is essential to its function, which is (partly) as an entry point for electrons (and by extension, energy) into the light-dependent reactions. Electrons caught and released by PS-II eventually reach PS-I where they are re-energized with energy from the P700 reaction center in PS-I. As the reader might have guessed, P700 is tuned to absorb 700nm light.

on many factors (species of plant, time of day, time of year etc.), but it is usually within the 400-700 nm waveband.

Action Spectrum

The *photosynthetic action* ocurring in a leaf can be found by measuring the rate by which CO_2 is consumed compared to the amount of light the leaf is exposed to[22]. The action spectrum shows the rate of photosynthetic activity at each wavelength of light within the spectrum (figure 2.3).

Absorptance and Quantum Yield

Absorptance (figure 2.4a) measures a surface's effectiveness in absorbing light, while quantum yield (figure 2.4b) measures the conversion rate of a light-dependent process (usually underneath the aforementioned surface) between the absorbed light and a desired product. The nuance between action and quantum yield is that the quantum yield measures production only against the radiance that was actually absorbed by that interface, while action does not take absorption into account.

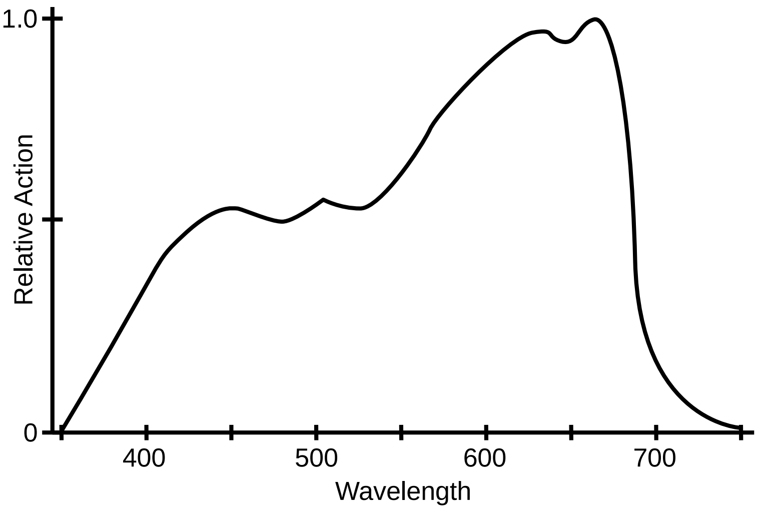
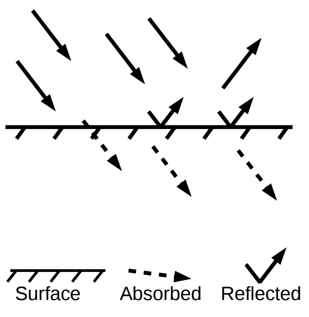


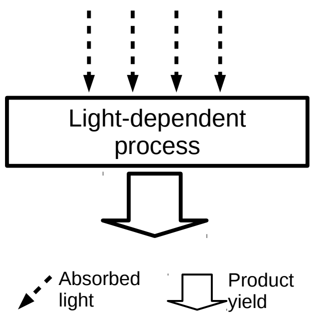
Figure 2.3: Mean of the relative action of 21 different species of plants[22].

2.2.3 Limiting Factors on the Rate of Photosynthesis

Emil L. Smith has shown that there are upper limits to the amount of light and carbon dioxide that can contribute to the rate of photosynthesis[29]. The effects of *Temperature* on photosynthetic reactions are convoluted as pointed out by Robarts et al.[27], but they explain that optimal temperature seems to be around 25°C.



(a)



(b)

Figure 2.4: Absorptance (a) and quantum yield (b)

2.3 Optimization Algorithms

2.3.1 Solution Space

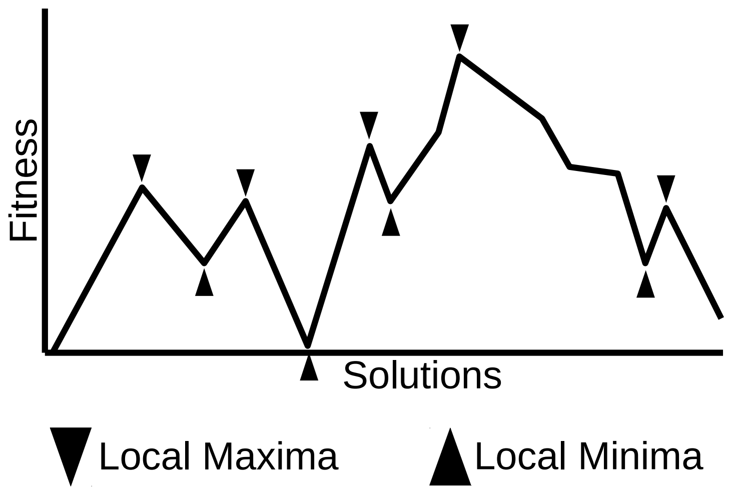


Figure 2.5: Solution Space: Solution spaces may have multiple *local* minima and maxima. The ultimate goal of a search algorithm should be to find the *global* maximum (or minimum), of which there can be only one.

In order to make a useful optimization algorithm, it is necessary to understand the *parameters* of the search, its *solution space*(figure 2.5) and how the two are connected. Solution spaces have as many dimensions as the search has parameters, along with one axis to evaluate the solution. Visualization gets tougher as dimensionality increases, but in essence, the solution space looks like peaks and valleys formed by solutions of different viablility. These peaks and valleys are known as *local maxima* and *local minima*, respectively, but for simplicity, only local maxima will be used from here on.

Neighborhood

Navigating the solution space efficiently requires a good definition of *neighborhood*. One definition of the neighborhood of a solution could include all the other solutions that are reachable by changing only a single parameter in the search (figure 2.6). If the search always follows the strictly best path through each successive neighborhood in its path, the same peak will be reached every time from that neighborhood.

`/home/bjornar/master-thesis/master-thesis/fig/technical-neighborhood.pdf`

Figure 2.6: Neighborhoods: This example search has two parameters A and B, and the neighbors of the current solution is the set of solutions which can be reached by only changing *either* A or B. The ways that algorithms define neighborhoods can vary greatly, and in the most general terms, a neighborhood is the set of solutions that the algorithm considers *close* to one another.

Solution Evaluation

The evaluation process depends wholly on the specific problem being tackled, but solutions are usually ranked by their *fitness* or *cost* function. Searches will either strive for the solution with highest fitness (global maximum), or lowest cost (global minimum).

Search Policies

There are two main *policies* that a search can use when it navigates the solution space - *exploitation* and *exploration* (figure 2.7). A strategy which favors exploitation will tend to favor the best solution in its neigbourhood, while more explorative strategies will sometimes take a chance that an inferior solution may eventually lead to a higher peak. Neither fully explorative, nor fully exploitative strategies are very effective, and algorithms land somewhere in between the two by stochastically choosing either exploration or exploitation at each step.

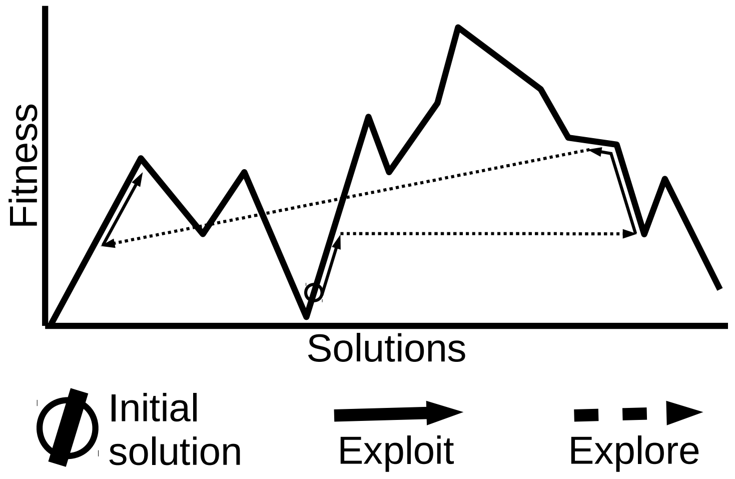


Figure 2.7: Exploitation versus Exploration: An exploitative policy will trawl the solution space, always choosing the best of its neighbors at each step. To avoid getting stuck on local peaks, some policies will make a jump (explore) across the solution space, though there's no guarantee that the jump will lead to a higher peak.

2.3.2 Hill Climber

Hill Climber (also called local search) is a naive search algorithm that uses a purely exploitative policy. Its inability explore new neighborhoods without the assurance of a better solution is what prohibits hill climbers to escape these local maxima. It is the most basic of optimization techniques, but its simplicity lends iself to modification and adjustment to the requirements of the application in which it is being used. For this application, Hill Climber will serve as a baseline to compare other approaches to.

2.3.3 Simulated Annealing

Simulated annealing (SA) is a probabilistic method to reach a close-to-desired heuristic for a given search problem. It uses a metaphor of cooling hot metal to gradually push the behavior of the search from exploration (flexible metal) towards exploitation (material solidifies) (figure 2.8).

The four ingredients needed for a successful SA algorithm are described by Kirkpatrick et al. [19]. The ingredients they talk are (1) the parameters, or dimensions of the search; (2) a function which generates new solution s by changing one parameter of the "parent solution" randomly; (3) a fitness (or cost) function that evaluates the search parameters of a given solution; and (4) the parameters that define an annealing schedule.

- Starting temperature
- Terminal temperature
- Relative change in temperature each time the "material" cools.
- Iterations per temperature change

These are the parameters which inform the annealing on how fast to cool the metaphorical metal, and when to stop doing so. The algorithm uses an acceptance probability (AP) function to decide whether it should choose a given solution or not. The AP compares a new solution to the current solution and decides how likely it is to choose the new one. Solutions with a higher fitness have a higher probability of being accepted, but this is also where the strength of SA lies. When temperature is high, the AP increases for *all* solutions, and the fitness is not as heavily weighed in the AP function. As temperature decreases, the emphasis on fitter solutions increase.

This dynamic leads SA to start by exploring the solution space in experimental leaps with the goal of discovering the best avenues to exploit later. This inevitably happens when the temperature approaches its terminal threshold. The speed at which the material cools is percentage based, which means it starts out high, but tapers off gradually, giving SA time to exploit the chosen avenue.

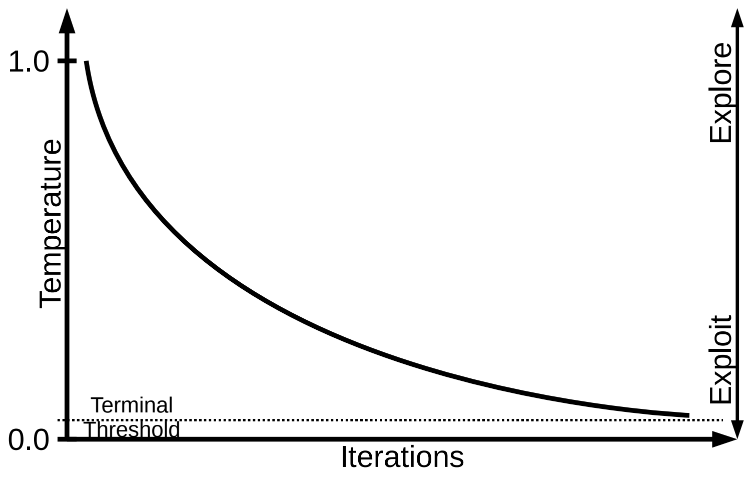


Figure 2.8: Illustration of how the policy of simulated annealing changes gradually from exploration to exploitation over time.

2.4 Components for and Consideration in Indoor Gardening

2.4.1 Microcontroller

A microcontroller is a small integrated circuit containing a CPU, memory and a set of pins that can interface with various types of peripheral components. The input and output across these interfaces can be programmed, and this makes microcontrollers well suited as prototyping hardware, and serving as controllers in larger systems. Arduino is a developer that focuses on open source entry level microcontrollers.

Arduino

Arduino boards are programmed in a dialect of C++, and the official code libraries are extensively documented with tutorials and examples which teach the relation between software and hardware. Their code base is open source so anyone can contribute as long as their code meets the criteria [3], which are themselves regulated by the community.

Code may be uploaded to the microcontroller using the Arduino IDE. The structure of Arduino programs revolve around two functions: `setup` and `loop`. The former is used for initialization and it runs immediately after uploading code or rebooting the Arduino. The `loop` function runs continually, and is where the work gets done.

The digital pins can function as either input or output. When the mode is set, `digitalWrite` or `digitalRead` can be called to communicate with the peripheral connected to that pin. Most of the analog pins are only for input, which is accessed by `analogRead`. Unless the board has special analog output pins, digital pins have to serve as proxies instead(figure 2.9). The `analogWrite` function initiates a steady generation of the specified duty cycle which continues until it is explicitly stopped.

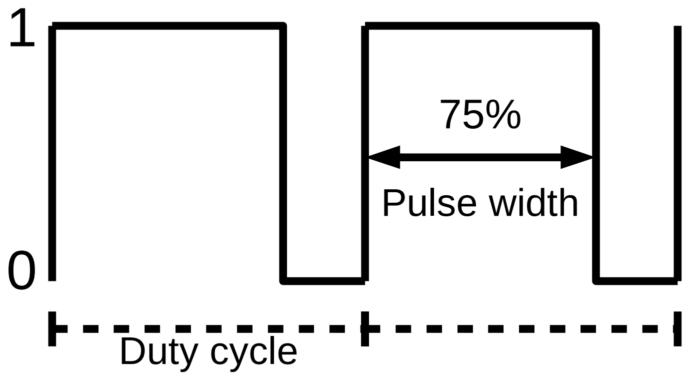


Figure 2.9: Pulse Width Modulation is a method that allows digital signals to emulate analog behavior. The pulse width is the portion of each duty cycle that the signal is 1. The signal is set to 0 for the rest of the cycle. The average voltage over the duty cycle can be used to adjust the brightness of an LED for instance.

Some boards also come equipped with a pair of SCL and SDA pins to interface with I^2C (explained later) and TWI (two-wire interface) devices. The Wire library contains abstractions like `beginTransmission` and `endTransmission`, `read` and `write` to enable implementation of a communication protocol.

The Arduino models Mega 2560, Uno and Due were compared (table 2.1)

2.4.2 Single-Board Computers

Single-board computers (SBC) are so named to differentiate them from multi-board computers, which are ubiquitous. These tiny computers can be found implemented as control units or communication interfaces in embedded systems, or on the workbench of a hobbyist in the maker community.

The applications of SBCs intersect with those of microcontrollers somewhat, and it is common to include a set of GPIO pins in their design. However, the strengths of an SBC lies in the higher layers of abstraction such as visual and network interfaces (e.g. HDMI and WiFi).

Three different SBC models were compared: Raspberry Pi model 3 B, Tinker Board and C.H.I.P. (table 2.2)

Board Name	Mega 2560	Uno	Due
GPIO	54	14	54
PWM	15	6	12
Analog I/O	16/0	6/0	12/2
Rx/Tx	4/4[1]	1/1[1]	4/4[1]
I^2C	SDA/SCL	n/a	SDA/SCL
Operating Voltage	5V	5V	3.3V
Clock Speed	16 MHz	16 MHz	84 MHz

Table 2.1: The distribution of analog pins and general purpose input output pins (GPIO), some of which can use pulse width modulation. The Rx/Tx pins are also included in the GPIO count, and they are connected pairwise to UARTs.[1]One Rx/Tx pair is connected to the same UART as the USB cable, and should be only be used with this in mind.

Model Name	Raspberry Pi 3 B	Tinker Board	C.H.I.P.
Visual Interface	HDMI	HDMI	HDMI
Networking	WiFi/Ethernet	WiFi/Ethernet	WiFi
Storage	microSD only	microSD only	4GB on-board
USB ports	4	4	2
GPIO	17	28	8

Table 2.2: A comparison of the features of the Raspberry Pi 3 model B, Asus Tinker Board and C.H.I.P.[9]

2.4.3 CO_2 sensing

Diffusion and Sampling

The use cases for a CO_2 sensor depends on the application, but there are two main ways to approach measurement: *diffusion* and *sampling*. Diffusion measurement depends on the principle that gasses spread evenly in ambient air over time. The diffusion sensor can be placed anywhere in an open space (e.g. office space) and assume that its surroundings are representative of the space at large. Sampling sensors take samples the air continually via vacuum pumps and generally have quicker response times. This makes them more fit for applications where concentrations are expected to spike, such as experimentation or leak detection.

Measurement Methods

A common (based on impression of sensor market) method for measuring CO_2 concentrations is called *non-dispersive infrared* (NDIR, figure 2.10). *Dispersive* sensors segregate the light within the needed waveband with a prism *before* shining it through the gas rather than filtering it afterward. Infrared sensing methods are susceptible to dust disturbing the measurement[34].

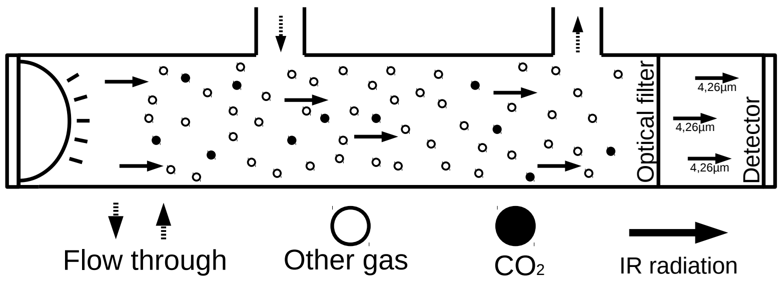


Figure 2.10: Non-dispersive infrared(NDIR) flow through sampling: At one end of the sampling chamber there is an infrared lamp. In the opposite end, there's an infrared detector covered by an optical filter. Air flows freely through the intermediary chamber and the molecules in the flowing gas will absorb certain wavelengths of light. The optical filter eliminates the light outside the absorbtion spectrum of carbon dioxide (~4.26µm[32]). The amount of radiation that reaches the detector can be used to approximate the concentration of CO2 molecules in the air.

Comparing Models

Here are three CO_2 models compared against each other: SEN0219, CozIR AMB and Telaire 6613 (table 2.3).

2.4.4 Climate Monitor

The temperature and humidity can be important to monitor, especially for outdoor greenhouses, where conditions are more volatile.

HYT 271 Digital Humidity and Temperature Module

This sensor is manufactured by Innovative Sensor Technology, and it can interface with an I^2C (explained below) bus through SDA and SCL pins. It measures temperature and relative humidity only after an MR (measurement request) command. When the internal processes are done, it stores the measurement as 4 bytes in the output register.

The master may invoke the DF (data fetch) command to read the output register. The first two bytes of the response containa a 14-bit value representing relative humidity (0-100%). The remaining two bits (bits 14 and 15) contain metadata. The second pair of bytes contains another 14 bits representing temperature (-40-125 degrees Celcius). The remaining two bits (bits 0 and 1) are not used.

2.4.5 Lighting

The considerations to take when choosing which type of lights to use in an indoor garden boil down to cost versus efficiency. The cost can be

Module Name	**SEN0219**	**CozIR AMB**	**Telaire T6613**
Sensing Method	NDIR	NDIR	NDIR
Sampling Method	Diffusion	Diffusion	Diffusion
Range(ppm)	**0-5000**	0-2000	0-2000
Accuracy	$\pm(50 + 3\%)$	$\pm(50 + 3\%)$	**400-1250: max($\pm$30, 3%)** **1250+: $\pm$(30 + 5%)**
Warm-up	3 min	**<10 sec**	10 min
Interfaces	Analog	Serial, Analog	Serial, Analog, I^2C
Calibration	Manual zero point	Automatic[1]	Automatic[1]
Conditions	0-50°C, 0-95 RH	0-50°C, 0-95 RH	0-50°C, 0-95 RH

Table 2.3: Viable CO_2 modules. [1] The automatic calibration routines in the CozIR and TelAire sensors calibrate to the lowest CO_2 concentrations they are exposed to within a given period of time and adjusts on the assumption that those concentrations are 400 ppm. In both cases, the automatic calibration can be disabled.

the power consumption (kWh) and/or the cost to purchase and install the lights. Efficiency should take into account the rate at which energy dissipates into heat instead of light, and also the portion of the light that falls outside the range of PAR. Even within PAR, some wavelengths of light are more efficient than others (as evident from the action spectrum): 'Red lamps are recommended for flower production and the blue for vegetative growing, whereas the mixed spectrum is for plants from start to finish.' (Hobby Hydroponics, Second Edition By Howard Resh)

Light Emitting Diodes (LED) are small individually, but make fully viable grow lights when fitted onto arrays. They are very flexible in how they are installed and this granularity is one of the advantages LED technology has over its alternatives. The user can design light fixtures with high precision in terms of focusing light output on those wavelengths with highest photosynthetic action. 'Estimations of the electrical energy conversion efficiency of a LED system for plant irradiation suggest that it may be as much as twice that published for fluorescent systems.' (Light-emitting Diodes as Radiation Source for Plants by R.J. Bula et. al.)

There are other alternatives, such as high intensity discharge (HID) and fluorescent lights, but none of them offer the same flexibility in configuration as LEDs.

Petunia LED board

This table shows the relevant points in the Petunia LED board specification (table 2.4).

Colour	Temp	Typical Wattage @700mA	LEDs	Forward Voltage	Radiance Angle
Hyper Red	656nm	7.35 watts	5	10-13V	80(±40)°
Warm White	3000K	13.02 watts	6	16.2-21V	150(±75)°
Deep Blue	455nm	2.17 watts	1	2.7-3.5V	80(±40)°

Table 2.4: Each Petunia board has 12 LEDs, 6 white, 5 red and 1 blue, distributed evenly around the board. LEDs of the same color are connected in series, creating distinct channels. The radiance angle can be sharpened by attaching a lens to the Petunia board[1].

2.4.6 Mechanical Components

Damper Control

Dampers are mechanical doors that can regulate the flow of air in ducts, containers or the like. The actuator that changes the state of a damper can be a servo. *Dynamixel AX-12* is a servo model with a well documented instruction set, and can be programmed to many different behaviors (e.g. wheel mode allows for a continuous turn).

The AX-12 servos use UART (explained below) for communication, but they are designed to receive and respond on a single line. Signals from the master are typically carried through a daisy chain of servos. They can be addressed individually (if given an address), or a general broadcast address can be used. In order to give the servos unique addresses, they need to be isolated from the daisy chain one by one so their address can be stored in an internal register.

Airflow

The *8412 N* model of fan was made by Ebm papst. The fan speed is controlled via PWM, and it has a tachometer which outputs the RPM of the fan[10].

2.5 Communication Interfaces

2.5.1 Universal Asynchronous Receiver-Transmitter

The UART is a hardware component that handles communication between one device and the UART on another device. Data is transmitted(Tx) serially (bit by bit) to the receiver (Rx) at the other end, which reassembles and stores the data in a receive buffer for that device to read whenever convenient.

Communication (figure 2.11) is asynchronous, so there is no need for a clock signal, but the *baud rate* (bits per second, usually) has to be agreed upon beforehand by both devices. The start and stop bits make up the *frame* of a byte. The *parity bit* tells the receiver whether that byte has an odd or

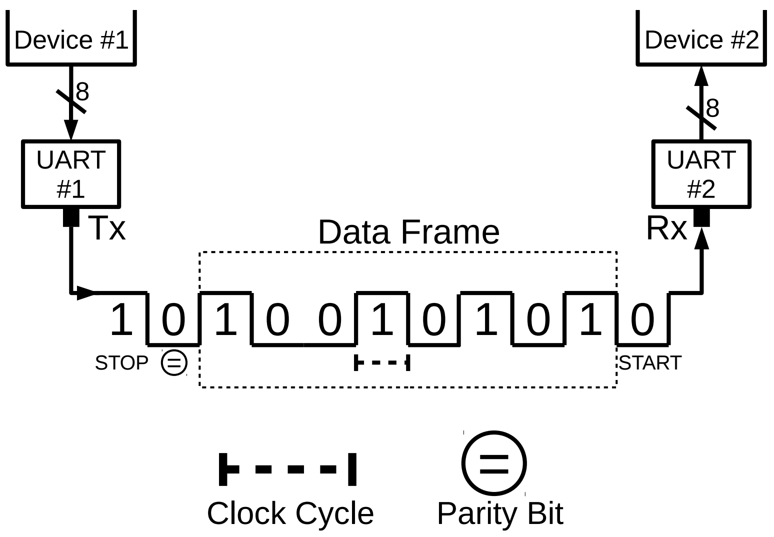


Figure 2.11: UART communication protocol: UART #1 frames the data coming across the data bus from its device and transmits it to UART #2 (LSB first), which unpacks the data and sends it to Device #2 via the bus. The default setting for UART packets is 8-bit frame, no parity bit and 1 stop bit (8-N-1 is a common shorthand). The clock cycle is defined by the baud rate, which has to be agreed upon before communication can take place. For two-way communication to be possible, an Rx/Tx line going the opposite direction is needed (omitted here for clarity).[6]

even number of 1s. This is used for error detection, akin to checksums in higher level communication. *The stop bit* is logical 1 and can be configured to transmit for 1, 1.5 or 2 times the normal cycle duration (determined by the baud rate). Extending the stop bit to more than 1 cycle increases the chance that a framing error will get caught, but increases the word length (bits per packet) [6].

2.5.2 I^{2}C

I^{2}C(Inter-Integrated Circuit) is a communication bus developed in 1982 by Philips Semiconductor that enables synchronized communication between multiple devices. While an I^{2}C bus may be configured in a variety of ways, this project uses the bare minimum of two lines; data (SDA) and clock (SCL). The SCL line is controlled by the master, and it dictates the bit frequency on the bus. Each time the SCL is low, SDA is ready to transmit a bit.

Because devices are open collector, both lines need pull-up resistors (lines rest high). The master initiates communication by pulling the SDA down while the SCL is high. Communication concludes when the master lets go (pulled up by resistor) of the SDA during clock high. These are the start and stop sequences, respectively (figure 2.11).

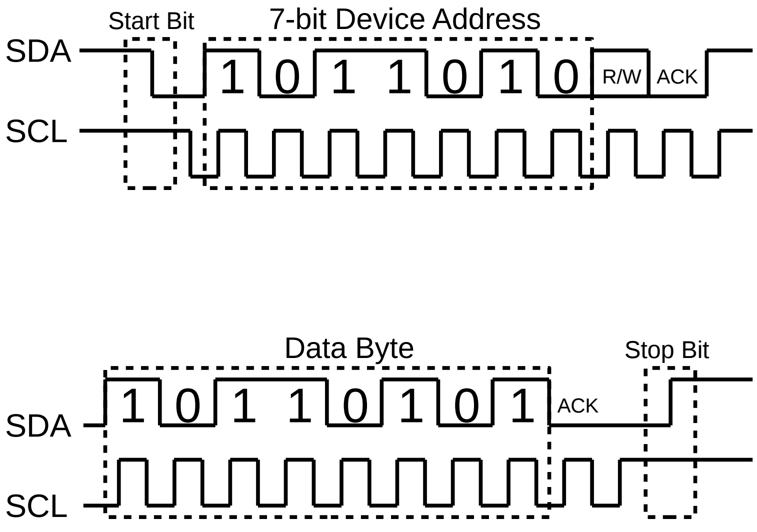


Figure 2.12: The master initiates communication with a special start bit (pulling SDA low while SCL is high), followed by a 7-bit address. The address should be unique to a specific device on the bus, and the 8th bit is used to signify whether the master intends to read from or write to that device. If the recipient device received the signal, then it pulls the SDA line low in acknowledgment. Data is read/written in much the same manner; 8 bits followed by an ack from the recipient. When a transaction is over, the master sends a stop bit (letting SDA go high while SCL is high).

- Write
  1. Send a start sequence
  2. Send the I^2C write address of the slave (7 bits + R/W bit low)
  3. Send the address of the internal register to write to
  4. Send data byte(s)
  5. Send the stop sequence
- Read
  1. Send a start sequence
  2. Send the I^2C write address of the slave (7 bits + R/W bit low)
  3. Send the address of the internal register to write to
  4. Send another start sequence
  5. Send the I_2C read address of the slave (7 bits + R/W bit high)

6. Slave holds the SCL low (clock stretching) until it's ready to serve the master
7. Read the data byte from the slave
8. Send a stop sequence

After a bit is transmitted on the SDA, the SCL is let go (high), then pulled down, ready for the next bit. Following every 8th bit received by the addressed slave, that slave pulls the SDA down to acknowledge (ACK) that a byte was received. A lack of acknowledgment from the slave raises an error, and the master must act accordingly. Devices that share the same I^2C bus are limited to 7-bit addresses because the 8th bit (LSB) is used to indicate a read or a write from the master.

2.6 Data Storage and Presentation

2.6.1 Relational Model

The relational model for databases structures its data by describing the relations they have to one another. Each *record* is a row in a *table* with *columns*. Each column represents either a characteristic of the record, a link to another record (*foreign key*), or a unique identifier (*primary key*). The *schema* of a relational database describes the inter-relationships of the tables (and its records, by extension)[26].

Structured Query Language

SQL is a language for interfacing with relational databases. Queries are written by filtering records based on the values in their columns, or their relations to other tables and records[30]. *Sqlite3* is a Python module that gives the ability to write wrappers for SQL queries in Python[2].

2.6.2 JavaScript Object Notation

JSON is a language-independent, human-readable format designed for data-interchange. It structures the data in familiar abstractions like *dictionaries* and *lists*, and the data is encoded in UTF-8 [18].

2.6.3 FrappeCharts

FrappeCharts is a module for JavaScript which is used to display dynamic charts in a web interface. It plots the data points from a JSON file and creates interactive and responsive models[13].

Chapter 3

The Framework

At the center of this project, there is a plant and a LED array, and the goal is to determine which wavelengths of light lead to the highest photosynthetic action, and to adjust the channels of the LEDs to approximate those wavelengths within a given power limitation. In order to achieve said goal, a framework was designed that consist of both hardware and software. This chapter will describe and justify the choices that were made during its implementation.

A link to the framework source code[1]. More detailed description appendix A.

3.1 Framework Requirements

3.1.1 Hardware

1. The framework should include a microcontroller to take care of low-level protocols
2. The framework should be headed by a single-board computer.
3. The single-board computer should issue all commands through the microcontroller.
4. The framework should include an enclosed space for plants to grow in.
5. The enclosure should be lighted by a pair of LED arrays.
6. The LED arrays should have channels with individually adjustable intensity.
7. The effect of LED heat production on the enclosure climate should be minimized.
8. Circuitry should be easy to trace from control/power source to individual components.

[1]https://github.com/bjornarprytz/led-action-framework/tree/master/source

9. There should be screw terminals to make it easy to switch out faulty wiring, and/or troubleshoot in the case of malfunctioning hardware without necessarily affecting the whole system.

3.1.2 Software

1. The framework should be able to control the spectrum and intensity of light inside the enclosure.
2. The framework should be able to visualize sensor data dynamically.
3. The framework should keep a database of sensor readings
4. The readings should be categorized by intervals and experiments in order to make data collection and analysis independent procedures.
5. The hardware components should communicate under well-defined protocols, to facilitate modular development.
6. The single-board computer should be able to execute high-level routines which parameterize experiments and intervals.

3.2 Hardware

3.2.1 Overview

The enclosure (figures 3.1, 3.2) is a 46x46x34cm^3 aluminium-edged box encased in plexiglass walls with a removable lid. 40 cm above the lid, the aluminium edges continue to create a fixture on which to mount the LED arrays. In order to allow heat to escape this fixture is not encased in plexiglass. The base inside the enclosure is covered by a metal grid which will be covered by soil. A thin space (~1 cm) between this grid and the plexiglass bottom collects excess water.

The components can be divided into two categories: *central* and *peripheral*. The peripheral components are specialized modules with a singular purpose in the framework, while central components are fewer, but have more complex functions. Most of the peripherals are either *inside* or *attached to* the plexiglass enclosure, which houses the plants themselves.

Figure 3.3 is an overview of all the circuits and components in this framework. It shows the connections from power to ground (with some simplification), and all communication lines between the microcontroller and the peripherals. The Raspberry Pi is not depicted, but it is connected to the Arduino with USB, which transmits communication as well as power.

3.2.2 Peripheral Components

There are two Telaire T6613 CO_2 sensors in this framework. One of them is positioned amid the growing plants for maximum exposure to their environment. The other CO_2 sensor is outside the enclosure to allow control for variance in the climate of the lab. There is a fan on top of the

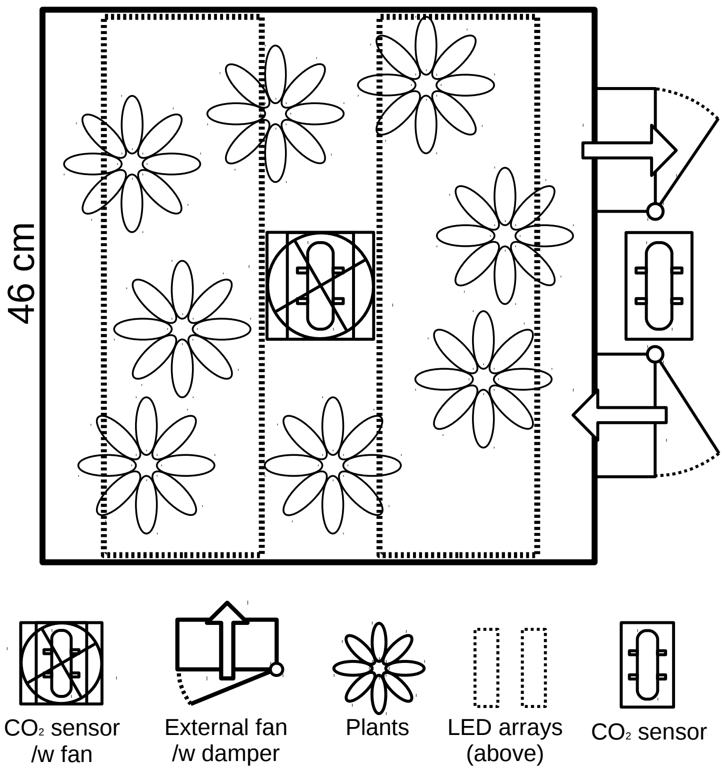


Figure 3.1: Top-down view of the enclosure: The internal CO_2 sensor is placed amid the plants, while the external one is between the dampers. Each sensor has a fan attached to keep the air surrounding it from going stale. Additional air flow is induced by the external fan mounted on the side of the enclosure, and they are opened and closed by servo driven dampers. The LED arrays are mounted above the enclosure, and their placements are indicated in the figure.

internal sensor to induce air-flow. The humidity and temperature sensor hangs mid-air above the plants.

Two holes in one of the plexiglass walls can be opened and closed by a pair of servo-driven dampers. A fan is attached to each of them. One of the fans is mounted to suck air out of the enclosure, while the other one blows air in. A pair of LED arrays are mounted onto the aluminum fixture 40 cm above the lid of the enclosure (figure 3.2) in order to minimize the heat generation in the enclosure. Each of the two LED arrays consists of six Petunia boards screwed onto an aluminium heat sink.

Assembling the Petunia LED arrays

The lighting technology that was chosen for this framework was LED, and the main reason for that is to be able to fulfill software requirement 1, which

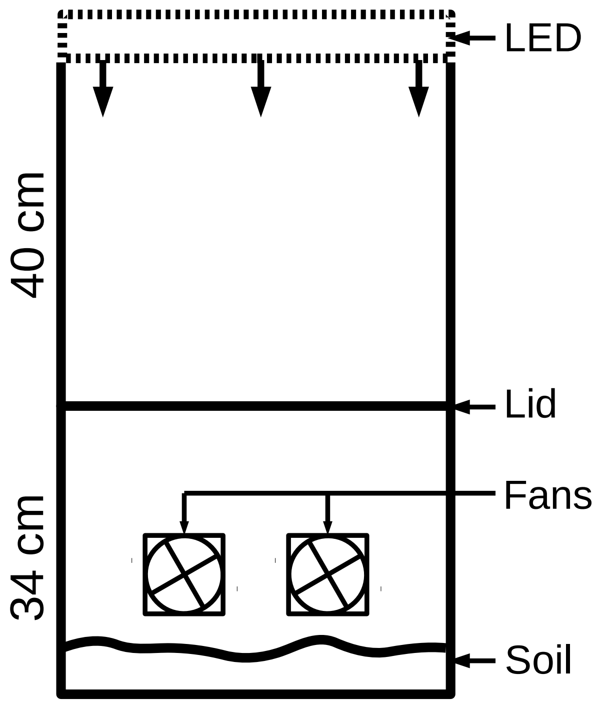


Figure 3.2: Sideways view of the enclosure: Each LED array is fastened to a heat sink which rests 40 cm above the enclosure on fixtures (no walls between them, so heat doesn't build up). The lid is made of plexiglass, just like the walls in the enclosure. The fans are shown here as a point of reference to Fig3.1

behooves the framework to control lighting. Petunia LED boards were chosen because they are designed for indoor gardening, and the boards are already organized with different color channels in mind. The three channels are *hyper red*, *warm white* and *deep blue*, but because the channels don't require equal amounts of power, they need different power supplies (table 3.1). The amount of current on each circuit is controlled by resistors.

The wire configuration of each LED array and its channels is illustrated in figure 3.4, and the power and control configuration is illustrated in figure 3.5.

Each Petunia board was fitted with a lens in order to avoid too much dispersal in the light. The LED arrays are mounted 40 cm above the enclosure in order to minimize the heat production inside it.

Channel	PSU	Current	LEDs	FV	Resistors
RED	36V	500mA	15	~34.5V	2x2Ω
WHITE	43.5V	500mA	12	~37.2V	1x10Ω
BLUE	12V	500mA	3	~9.3V	2x2Ω

Table 3.1: Using the forward voltage (FV) stated on the datasheet as starting points, the needs of each circuit were found using a multimeter and a secondary, adjustable power supply which throttled the current at 500mA. Resistors were added to fine tune the circuits to fit our available power supplies and to limit current.

Telaire 6613 CO_2 Module

This module was chosen for CO_2 because it had the highest stated measurement accuracy (400-1250 ppm: ±30 ppm or 3% of reading, whichever is greater) among the models that were considered. The air inside the enclosure is expected to fall below 400 ppm because of photosynthesis, and the T6613 datasheet fails to mention what the accuracy is below this concentration. The reason for this is likely that it has an automatic calibration routine (ABC logic) which treats the lowest measurement over a period of time as the new ambient concentration (400 ppm).

Because this model uses diffusion sampling (passive sensing), it was necessary to attach a fan to the internal sensor. The reason for doing so was to keep the air around the sensor from going stale, and to hopefully gain some of the advantages of flow-through sampling (faster detection of changes in the enclosure). The wires connect the T6613 sensors to power and control are illustrated in figure 3.6.

HYT 271

Climate control is done by the HYT 271 temperature and humidity sensor (figure 3.7). Its purpose is to be able to take climate into consideration if need be. The climate in the room where the enclosure is placed is quite stable, but this sensor could be very valuable in a more volatile environment.

This sensor was chosen for its high precision measurements. The 14 for each of its measurement values (humidity and temperature) allows for high resolution.

Mechanical Components

The dampers are controlled by a pair of Dynamixel AX-12 servos (figure 3.9). Two of the fans are attached between the damper doors and the enclosure wall, and they are mounted in order to blow air in opposite directions (one blows air in, while the other one blows air out). Another fan is attached to the internal CO_2 sensor (figure 3.8). A fourth fan may be added, but it has not actually been used in this project. All of the fans and servos are powered by the same 12V power supply.

Managing the Circuitry

To avoid long stretches of wiring, there are 5 break points where collections of wires are organized by *screw terminals*. Terminal (1) is located above the row of PSUs, and it routes power to all parts of the system, and finds common ground for the PSUs as well(Appendix B.2). Terminal (2) is located just below the PSUs and it routes LED ground via 3 MOSFETs, and the servo control signal(Appendix B.1). Terminals (3, 4, 5) route all sensor and fan communication wires along with power (5V and 12V) and ground for those components(Appendix B.1, B.3, B.4). Terminals (6, 7) route the channels as they exit the arrays, before they are led to the resistors (Appendix B.5, B.6).

- Sensor communication wires were coded yellow/green for Rx/Tx (UART) and SDA/SCL (I^2C).
- Voltage sources - red (with some exceptions)

  The 12V source for the fans was coded pink.

  The source for each Petunia channel was colored to match it.
- Fan control - purple
- Servo control - dark blue
- Ground - black

3.2.3 Central Components

Arduino Mega 2560

All of the peripheral components are controlled directly by an Arduino Mega 2560 via various communication interfaces (UART, I^2C bus, PWM). Out of the 54 GPIO pins, only 13 were used, but the high requirements for UART interfaces made the Arduino Mega 2560 the choice over the more lightweight Arduino Uno. The much higher clock speed of the Arduino Due compared to the Mega was not needed for this project, and the 5V operating power of the latter seemed more important when interfacing with a wide range of devices.

Raspberry Pi 3 Model B

The Raspberry Pi 3 Model B is the head control unit for this framework. It is connected via USB to the Arduino, which transmits communication as well as providing power to the microcontroller. The RPi is also connected to a screen via HDMI, which was useful for installation and debugging, but most of the development was done through secure shell (SSH). This made WiFi essential for the workflow. In addition to the aforementioned reasons, the RPi was chosen for its multitude of tutorials on installation and usage with Arduino boards.

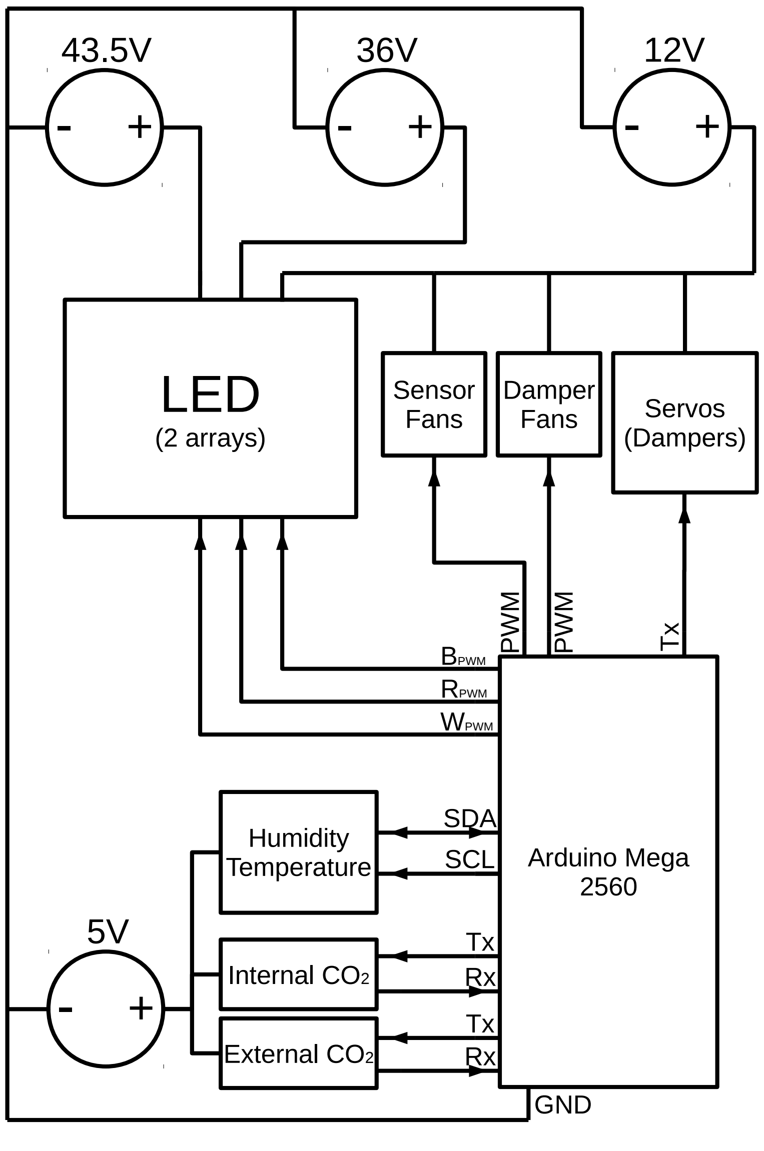


Figure 3.3: A high-level overview of the circuits of all the peripheral components in the framework. All control I/O to and from the Arduino is summed up here. All the peripherals are connected to ground, but these lines have been omitted in the figure to avoid clutter.

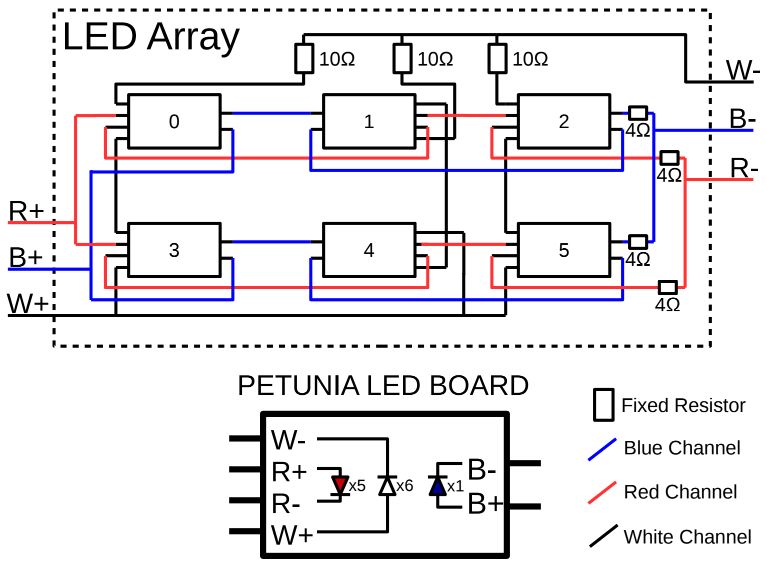


Figure 3.4: LED Array circuit overview: The Petunia boards have 12 LEDs on them; 6 white LEDs in series, 5 red LEDs in series and a single blue LED. **The red channel** is spread across two parallel series: (0,1,2) and (3,4,5). Each series consists of 15 LEDs and two 2Ω resistors. **The blue channel** is spans the same boards ((0,1,2) and (3,4,5)), but each series only contains 3 blue LEDs and two 2Ω resistors. **The white channel** is arranged into three parallel series: (3,0), (4,1) and (5,2). Each of them contains 12 LEDs and and a 10Ω resistor.

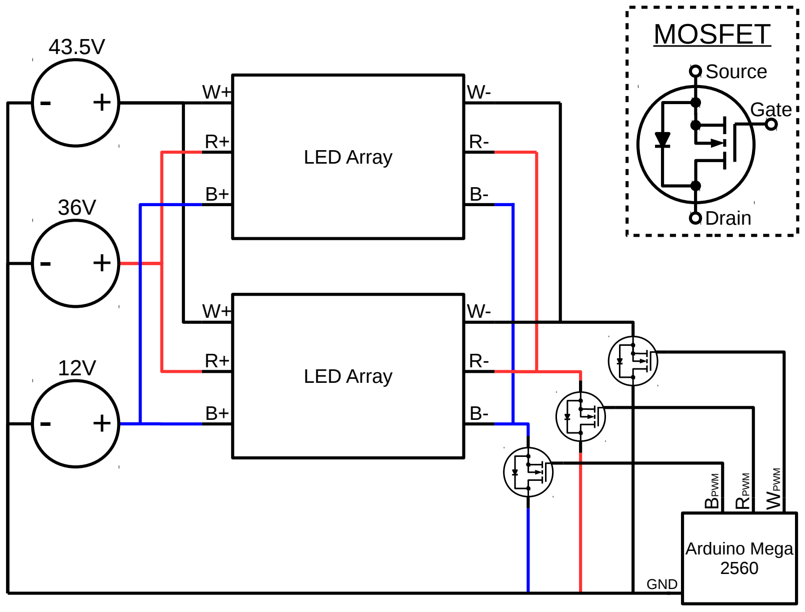


Figure 3.5: LED Array with power source and control. The framework contains two equivalent LED arrays. There are three pairs of channels with corresponding colors. Each pair is joined in parallel and powered by separate sources. Before connecting to ground, each channel goes through a MOSFET. An Arduino controls each channel with PWM on the MOSFET gate.

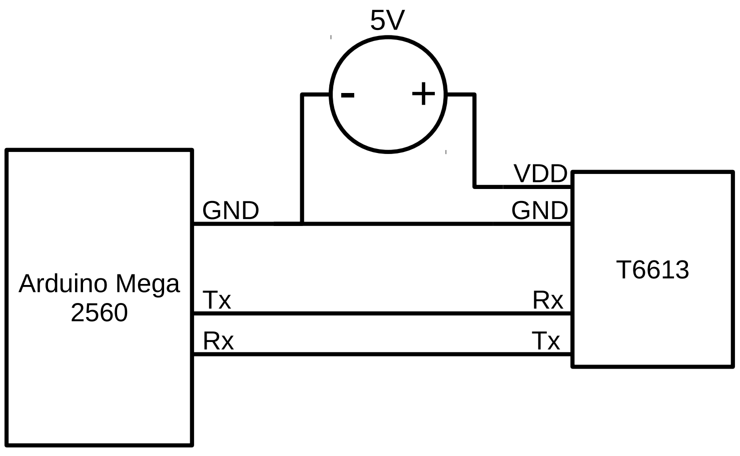


Figure 3.6: The Arduino and CO_2 sensors interface via UART (Rx/Tx).

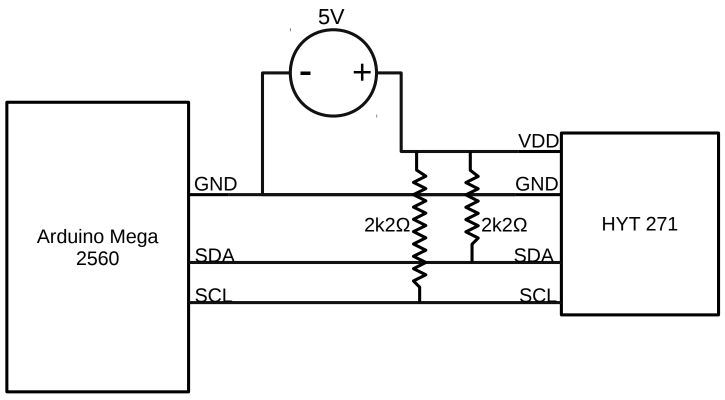


Figure 3.7: The temperature and humidity sensor is connected to an I^2C bus where the Arduino is master. The sensor has open collector output, so there are pull-up resistors on both I^2C lines.

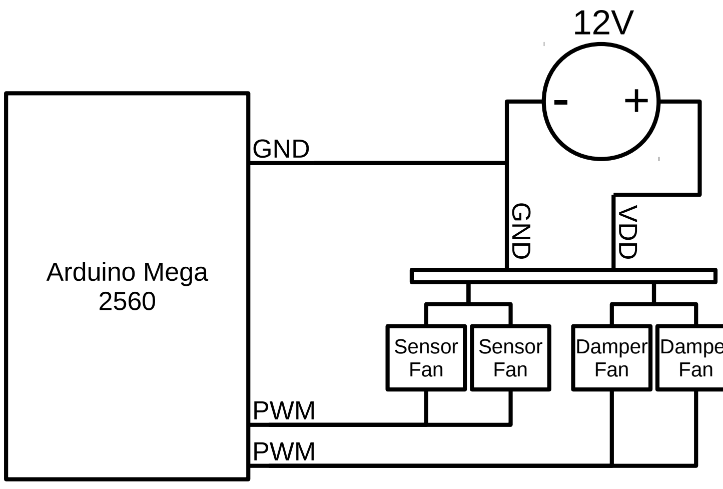


Figure 3.8: The Arduino uses PWM to control the fan speed. To limit complexity, there are only two control pins for four fans: one pin shared by the sensor fans, while another pin controls the damper fans.

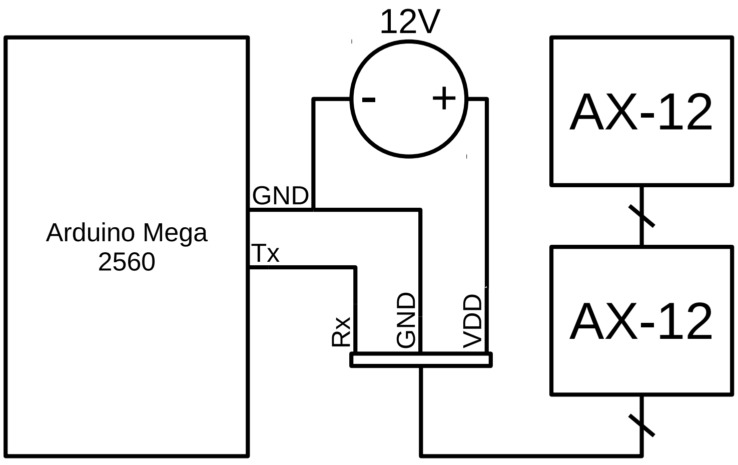


Figure 3.9: The two dynamixel AX-12 servos are connected in a daisy chain, and powered by a 12V power source. They are configured to not respond to commands from the Arduino, so communication is a one-way affair.

3.3 Software

3.3.1 Overview

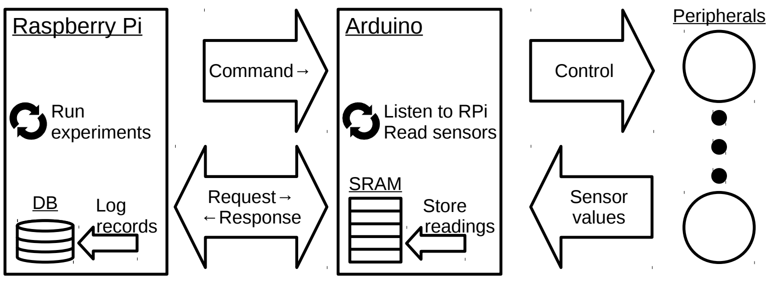


Figure 3.10: The Arduino periodically (every 10 seconds) stores readings from its sensors in memory. When the RPi makes a request, the microcontroller fulfills it with a stored value. The values are logged in a local database, along with a timestamp. This timestamp is made when the record is logged, so its relation to the reading is only an approximation (within 10 seconds of error).

The software portions of the framework are located in the Raspberry Pi (RPi) and the Arduino. The *RPi* acts as master of the *Arduino*, which in turn is master of the *peripheral components*. There are interfaces between these three groups, and in each case, its master initiates all transactions over that interface(figure 3.10).

The RPi runs Raspbian Strech Lite, which is a desktop-less operating system based on Debian [11].

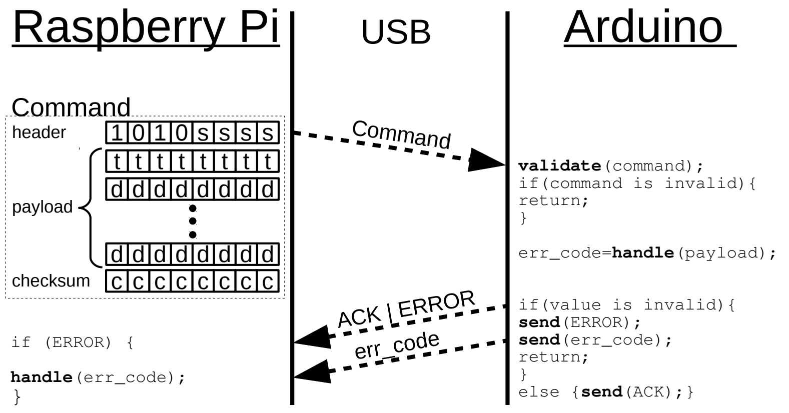
There is a fourth layer, though it resides within the RPi itself, and this is the *presentation layer*. Its purpose is to make development and general troubleshooting quicker and less tedious. In addition to workflow, the visual presentation makes it easier to show and discuss the framework with outsiders.

3.3.2 Serial Communication Protocol between Arduino and Raspberry Pi

Whenever the Arduino finds data in the receive buffer, it tries to validate the data as a complete packet. If at any step the Arduino finds an error in the data, it aborts validation and gives no response. Validation has two steps: (1) Check magic number (1010) in the first 4 bits of the packet header, and extract the message size from the remaining 4 bits and (2) Read the rest of the packet and verify it with the checksum at the end of it.

Once the packet is validated, the payload can be handled. The first byte of the payload contains the instruction, and this informs the Arduino on how to proceed. Most commands require parameters to execute, while requests don't require any information beyond the instruction.

The Arduino-RPi protocol is currently limited to 256 unique instructions, because only one byte in the payload is programmed to address the instruction set. This set encompasses both requests(figure 3.12) and commands(figure 3.11).



```
if (ERROR) {

handle(err_code);
}
```

```
validate(command);
if(command is invalid){
return;
}

err_code=handle(payload);

if(value is invalid){
send(ERROR);
send(err_code);
return;
}
else {send(ACK);}
```

Figure 3.11: RPi sends a command for the Arduino to execute: After packet validation, the command is handled by the Arduino, and if an error is caught, then it is sent after an error signal. If no errors ocurred, an acknowledgment is sent back to the RPi, and that concludes the command.

Dampers [open/close]

This instruction takes one parameter which can be either 1 or 0, commanding the Arduino to open or close the dampers. The specific values needed for the servos to put the damper doors in an open or closed state have been hard programmed into the Arduino function that sends the signal. The reason for this is to remove some unneeded complexity from the RPi-Arduino interface.

LED [red, white, blue]

The LED command takes three parameters, one byte for each channel (red, white and blue, respectively). The Arduino adjusts the PWM frequency on the three channels in accordance with the input.

Sensor/Damper fans [speed]

The commands for controlling fan speed adjusts the PWM frequency to the internal fan PWM speed control. The commands for sensor and damper fans are two separate instructions, and they require one parameter, which is the desired fan speed.

Calibrate CO2 [target_msb, target_lsb]

The calibration command takes the calibration target spread between two bytes as parameters. The Arduino goes through the steps of single-point calibration for both sensors: Turn ABC logic off; set target for calibration (CO2 concentration around the sensor); verifies that the sensor got the correct target; start calibration; wait for the sensor to finish its internal calibration routine.

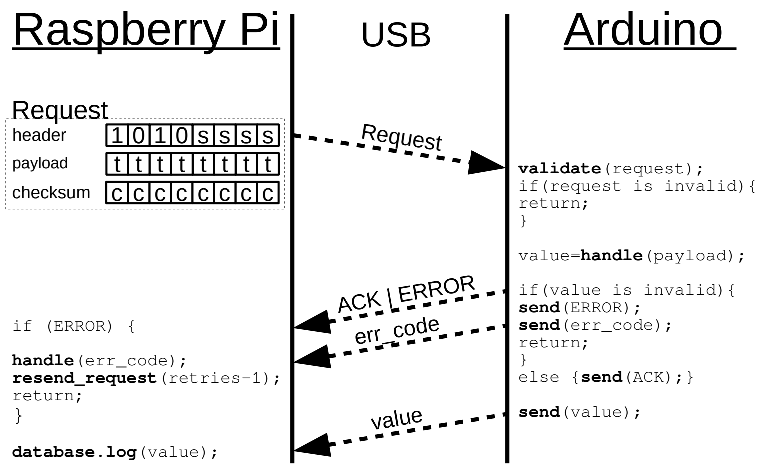
If any of the steps fail, the Arduino aborts the procedure and returns an error to the RPi.

Warm-Up CO2 []

This command does not require any additional parameters because its only task is to initiate a pre-programmed warm-up sequence in the CO2 sensors. As long as a sensor is warming up, the Arduino will give an error instead of the expected value from that sensor if it is requested by the RPi.

Requests

All the requests(figure 3.12) are similar in that they don't require any additional parameters. They prompt the Arduino to respond with the internally stored value for that sensor. It does not directly forward requests from the RPi to the sensors. There are four types of request instructions: Internal CO2, external CO2, temperature and relative humidity.



```
if (ERROR) {

handle(err_code);
resend_request(retries-1);
return;
}

database.log(value);
```

```
validate(request);
if(request is invalid){
return;
}

value=handle(payload);

if(value is invalid){
send(ERROR);
send(err_code);
return;
}
else {send(ACK);}

send(value);
```

Figure 3.12: RPi requests a value from the Arduino: If the request packet is valid, but there is an internal error Arduino-side, it responds with an error signal, followed by an error code. If the requested value is found valid, an acknowledgment is sent from the Arduino, followed by the value itself. If the transaction is completed successfully, the RPi logs the value in its internal database, otherwise, it will retry the request a number of times.

Error Detection

The Arduino may encounter errors when communicating with its sensors, and in most cases, it will just restart the communication, but sometimes it is necessary to raise the error all the way up to the Raspberry Pi. The error codes were implemented ad hoc to debug some of the more complicated routines like warm-up and calibration of the CO2 sensors. During runtime, they can indicate when a sensor is in warm-up for instance.

3.3.3 Arduino and Peripheral Communication

Petunia LED array

As has been explained above, the LED arrays are divided into 3 channels. The channels are gated by MOSFETs, which lets the Arduino adjust the brightness of each channel using the `analogWrite` function. Such adjustments are initiated by the RPi and have a resolution of 8 bit.

CO_2 Readings

The interface between the Arduino and the Telaire 6613 CO2 sensors utilizes a library called SoftwareSerial[4], which enables a selection of digital pins to behave like UARTs. The protocol that is implemented transmits control signals via (virtual) UART, and retries transmissions until a response is received from the sensor.

Humidity and Temperature Readings

The interface with the HYT 271 humidity and temperature sensor is standard I^2C. The sensor device address on the I2C bus is specific to the model, and can be found on the data sheet (0x28 for the HYT271).

Servo Control

The Arduino uses a single Tx pin to transmit to the AX-12 servos (they connected in a daisy chain). The servos are configured not to respond to the Arduino control signals (there is no Rx line from the Arduino to the servos). The Arduino uses the SoftwareSerial library to interface with the protocol developed by Dynamixel.

3.3.4 Database Interface

A simple relational database(figure 3.13) has been implemented to store readings that are taken in the enclosure. It is organized such that all readings are part of an interval, which are part of an experiment. It was designed to allow for cross-experiment analysis. For instance, a user could query the database for all intervals with a particular setting and try to determine the viability of that setting. The database runs on SQLite, and the code is written with the Python module sqlite3.

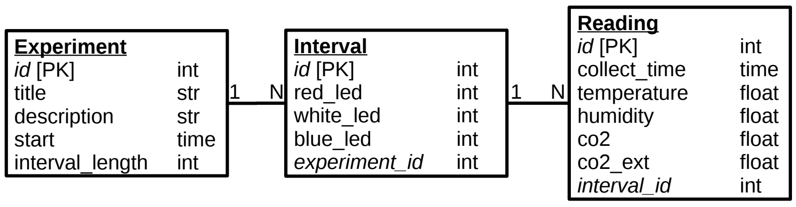


Figure 3.13: This schema describes a relational database that organizes the readings made by the sensors. Each experiment can span any number of intervals, and the length of those intervals is stored in interval_length (seconds). The intervals may use different LED settings, and they are stored in the database, along with a reference to the experiment that it belongs to. Each reading has sensor data and a timestamp.

This framework has implemented a thin interface (most functions are just wrappers for SQL queries) with the relational database. The basic function `execute_SQL` needs an SQL string and parameters to return search results. Using this as a base, more complex search queries can be written, though it requires some knowledge of SQL.

3.3.5 Presentation

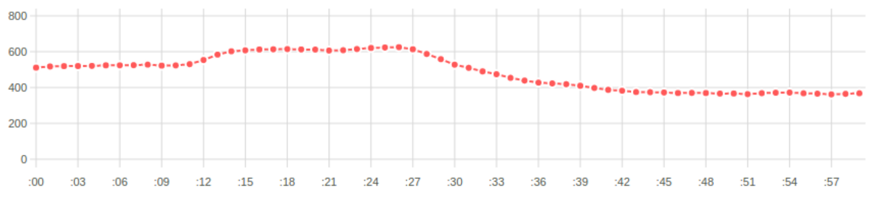


Figure 3.14: A screenshot of a graph showing the development of the CO_2 during an hour. Made with FrappeCharts[13].

Code was written to dynamically present sensor readings in HTML. It was written partly in JavaScript, partly in Python. The Python code queries the database and formats readings into JSON. The JSON is stored on disk and picked up by the JavaScript code (using FrappeCharts) on demand. The RPi runs a local server which can be accessed via SSH. The visual presentation displays readings every minute over the last six hours, as well as hourly readings over the last week. Figure 3.14 is a screenshot of the visual representation that was created with FrappeCharts.

3.4 Adding new components to the framework

One of the goals of this framework is to make the process of adding components to it fairly smooth. The steps for adding a sensor are as follows:

1. Determine which interfaces are available to the sensor, and connect it to the appropriate pins on the Arduino.

2. Write code for reading from the sensor, and upload it to the Arduino, preferably as a custom library (put the source and header files in a folder with the same name in /arduino/libraries/).

3. Update the RPi-Arduino protocol to allow the RPi to request readings from the new sensor. This is done by assigning the new instruction to an unused 8-bit number (256 unique), and this instruction table has to be consistent on both the Arduino and RPi.

4. Update the database schema and write the higher level functions which extract the necessary values from it.

Chapter 4

Methodology and Evaluation

4.1 Experiences While Developing the Framework

4.1.1 Assembling the Petunia LED arrays

After determining the which specific components were needed in each circuit (table3.1), 12 Petunia boards were fit onto a couple of aluminium heat sinks with, 6 boards on each (table3.4). A thin layer of thermal paste was coated onto the back of the Petunia boards before screwing them onto the heat sinks.

The resistors for each circuit were soldered together and fit onto a separate heat sink. Molexes(pin-and-socket wire collectors) were made to carry the circuits from the power supplies to the arrays and from the arrays to the resistors. From the resistor rack, the circuits congregate back into three channels and are led through one MOSFET each before connecting to ground.

4.1.2 Installing the External CO_2 Sensor

It was quite late in the development of the framework before the external sensor was deployed. Even after the plants had grown to full size, the CO_2 levels steadily increased day by day. When troubleshooting the error, another T6613 sensor was brought in to compare against, and it became evident that the original sensor had built up a 135 ppm offset because of the ABC logic. This routine was disabled (by writing to an internal register) as soon as it was discovered, and both sensors were single point calibrated using a third CO_2 sensor (Testo 535[8]).

Once both sensors were up and running, we started seeing quite large spikes in the sensor readings (sometimes above 3000 ppm, which is above even the spec of the sensor). The diagnostic process took a few days, but after testing for some potential causes (infrared interference, nearby ventilation) and working around the problem, we found out that the Arduino (which was supplying the power for the sensors at that time) was not delivering enough current for the sensors. The issue was solved as soon as we connected all the sensors (including the humidity and temperature sensor) to a dedicated power supply.

4.2 Measuring the Rate of Photosynthesis

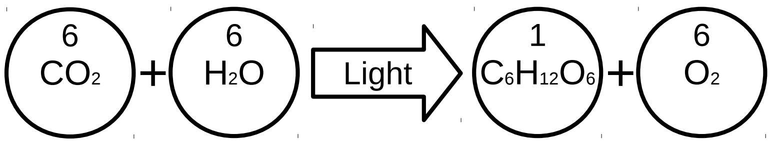


Figure 4.1: Illustration of the reactants and products of photosynthesis. It takes 6 CO_2 molecules and 6 H_2O molecules to synthesize one molecule of carbohydrates and 6 O_2 molecules

The definition of photosynthetic action is the rate at which carbon dioxide is taken up divided by the rate at which energy received by the leaf[22]. Energy received by the leaf will not be measured in these experiments, but energy consumption will be used as an approximation. The validity of this approximation remains to be seen.

4.2.1 What Sources of Disturbance are there in the Experiments?

To identify where *static* (fairly stable over time) and *dynamic* (unpredictable) errors occur in the measurement process of an experiment, we'll consider the aspects of a *perfect measurement* of photosynthetic action:

- The enclosure is air-tight, and only the plant itself can alter the contents of the internal air.
- The plant has access to ample (i.e. not limited by) nutrition and hydration.
- The only sources of light affecting the plant are the LED arrays, and all of their radiance is collected by reflective walls on the inside of the enclosure.
- Air flow is good and constant inside the enclosure, and the internal CO_2 sensor is well placed in order to have a representative sample of air.
- The CO_2 sensor is well calibrated and is not disturbed by other equipment.

Air Control

If there is a leakage in the enclosure then the CO_2 concentration would rise against the pressure put on it by photosynthesis. In the case of our enclosure, the extent of the contamination would be unknown, as would the composition of the contaminating air. This is likely a dynamic error because of how unpredictable the air flow and climate inside the enclosure is.

Water and Nutrition

The effect that water and nutrition has on a plant's ability to do photosynthesis is not taken into consideration in these experiments. The plants were not given any supplemental nutrition, save for what was in the soil they were planted in. They were given about 1,5 liters of water each day, but this was not measured with any particular accuracy. If malnutrition was consistent and it started affecting the plant's health an error in this area would probably manifest as static.

Light Sources

The light source is placed about half a meter above the plants, and a reflective surface was fastened around the support fixtures of the LED arrays. Their purpose was mainly to protect the eyes of students working in the room where the enclosure was placed, and no further effort was made to trap the light inside with the plants. This leads to two different sources of disturbance in the measurements of photosynthetic action: First, the more light that is allowed to escape the enclosure, the worse the power consumption of the LED arrays estimates energy received by the plants; secondly, the lack of screening from the outside introduces unaccounted-for rays of light which help drive photosynthesis.

The sum total of this disturbance is difficult to quantify. The room where the enclosure is placed has motion sensors which control the lighting, so this error should be considered dynamic, but because it is somewhat controllable it could be treated as static if experiments were only carried out while lights are on (or in the middle of the night).

Representative Sample

A representative sample is essential to a good measurement, and to achieve this, it is important to keep stirring the air. A failure to do this can lead to sensors stuck in pockets of stale air, unable to sense the changes in the enclosure as a whole. A shortcoming in this aspect would lead to our diffusion (passive) sensors having slow response times, and unnecessarily lengthy experiments. This is a dynamic error, because of the unpredictability of gas.

Sensor Handling

Calibration should be done with a reference gas with a known CO_2 concentration. We did not have such a gas at hand, so the internal and external T6613 sensors were put next to each other and single-point calibrated using a separate sensor. Calibration errors are static errors because it manifests as a persistent offset on measurements and don't change over time.

If sensors are operated badly, it can manifest as dynamic errors. In early tests, the sensors had an insufficient supply of current, and this lead to measurements spiking sporadically. Dirt building up on the sensor seals (or getting into the sampling chamber) could become an issue if it goes far enough, though it would be likely to cause a static error, disturbing the internal measurement.

4.2.2 Accounting for the Limiting Factors of Photosynthesis

Light being a limiting factor in photosynthesis means that the task of finding the optimal brightness for a given plant requires more finesse than simply turning all LEDs to maximum intensity. While the rate of photosynthesis might not decrease (unless heat generation becomes a factor) with excess lighting, energy will certainly be wasted.

External light has the potential to dampen the results of the experiments described here as it would bring general light intensity closer to the plateau where the plant stops benefitting from excess light. This effect would essentially "raise the floor", and decrease the potential benefit of LED lighting. The spectrum and intensity of the external light certainly matters, but if the sum total of irradiance doesn't saturate the plant, this "light pollution" may not hamper the experiment (beyond being a source of static error).

Ambient temperatures are controlled by the climate control in the building where the enclosure is installed. This means that the temperatures during experimentation should stay quite stable at 24°C (±1). A slight temperature increase (~30°C) has been seen from the LED arrays when they are at full effect and there is no active ventilation. With no particular grounds to say so other than intuition, the 23-30°C range should be within operating limits of the plant (Raphanus Sativus L.), and we have made no observations that suggest otherwise.

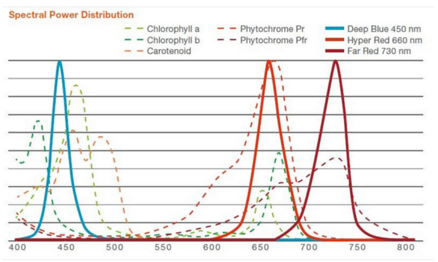
4.3 Simulating the Enclosure

The goal of the simulation is to discover a set of good *settings* for the physical experiments that are carried out in the enclosure. A setting is a vector of three numbers between 0 and 1. Each of those numbers describes the strength of the output of the respective channels. The energy cost of each channel is different, and the reason for this will be explained further on.

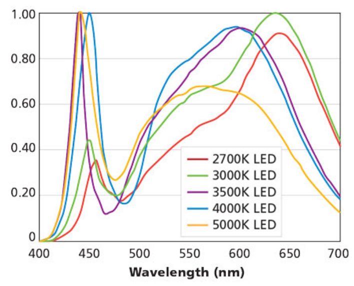
The meandering internal processes of photosynthesis means that time would be a major constraint when doing a real-time search with plants. The advantage of having a simulation step is that it can utilize the speed of a CPU to try many settings. Even if the gap (reality gap) between simulation and reality is large, the assumption is that it will be better than starting with a random setting.

There are some aspects that the simulation does not take into consideration

1. The action spectrum of a plant may change over time as it goes through its life cycle (growing, flowering etc.)



(a) The deep blue(peak ~450nm) and hyper red (peak ~660nm) power distributions were used estimate the output of the blue and red channels of the LED arrays.[16]



(b) The 3000K (peak ~645nm) power distribution was used to estimate the white channel output of the LED arrays.[20]

Figure 4.2: The process of creating the data points to simulate LED output required tracing the lines of the graphs and estimating their value at each data point.

2. How the time of day affects the plant's behavior
3. The possibility that plants may have a saturation point where it can no longer utilize light of a particular wavelength. The only limiting factor is energy consumption.
4. The Emerson Enhancement Effect - The synergy effect of stimulating two photosystems at the same time.

We tried searching for the optimal setting with both simulated annealing and a hill climber with random restarts.

4.3.1 The Parameters of the Simulation

To simulate the cost and effect of the LED arrays, we had to take into consideration the three different channels (red, white and blue), the output light spectrum and power consumption (wattage) of each of them. The PAR range treated as slightly wider than normal (from 350 to 750 nm). The data is divided into 17 data points with 25 nm between each point, and the reason is that the data we have on the action spectrum of Raphus sativus was presented in this form (table 4.1).

Output Spectrum

The *Power Distribution* for each channel is a vector that was made by manually tracing the graphs from the LED manufacturer (figure 4.2). For each channel in the LED arrays, the output of that channel is calculated as follows:

$$ChannelOutput = PowerDistribution * ChannelStrength$$

A setting describes what the strength of each channel is (e.g. This setting [1,1,1] would turn all channels up to maximum strength.), and the total output spectrum of the LED arrays as a whole can be found by adding the output vector for each channel together:

$$LEDOutput = Output_{red} + Output_{white} + Output_{blue}$$

Plant Response

The process of estimating plant behavior is a bit more straightforward as the data points were gathered from a paper([22], table "Relative Action of Growth Chamber Plant Species") which have already examined the action spectrum of the plant that we were using for our real-world experiments. It is a species of radish called Raphanus sativus.

Wavelenght (nm)	Mean	Raphanus sativus
350	0.09	0.24
375	0.28	0.48
400	0.43	0.50
425	0.52	0.52
450	0.54	0.53
475	0.51	0.53
500	0.56	0.56
525	0.55	0.52
550	0.61	0.58
575	0.75	0.70
600	0.88	0.85
625	0.95	0.92
650	0.96	0.98
675	1.00	1.00
700	0.42	0.41
725	0.09	0.08
750	0.02	0.02

Table 4.1: The data points were taken directly from this table (Raphanus sativus) and used as a measure of plant response in the simulation Source: [22], table "Relative Action of Growth Chamber Plant Species".

Energy Consumption

The total energy cost of a given setting is based on the relative wattage of each channel and it is calculated thus:

$$EnergyCost_{Watts} = Strength_{red} * \mathbf{3.4} + Strength_{white} * \mathbf{6} + Strength_{blue} * \mathbf{1}$$

The constants have been derived from the datasheet of the Petunia LED board[1] based on typical wattage at 700 mA. The wattage is normalized to blue, because it's the channel with the lowest cost.

The Reward Function

The function that evaluates each setting multiplies the LED output vector with the action spectrum vector. The result is the *plant response vector*, and it is an estimation of the yield of photosynthesis under the given setting across the PAR spectrum. The fitness of the setting is arithmetic mean of the 17 data points in the plant response vector. The reason for this last simplification is that the search algorithms will compare the fitness of many solutions, and having a scalar value makes the search more efficient.

$$PlantResponse = LEDOutput * ActionSpectrum$$

$$Fitness = mean(PlantResponse)$$

Energy Allowance

In order to take energy into the equation, we set a static amount of energy that settings were allowed to spend. This energy allowance was set to 50% of maximum wattage.

Finding Hypothetically Viable Settings

In order to search the solution space created by our simulation, we employed two different algorithms; a simple hill climber with random restarts, and simulated annealing. The goal was not to compare the two, but just to try different approaches to have more chance of success.

4.4 Experiments

Around 150 individual seeds of radish (Raphanus sativus L.) were planted in 21.16 dm^2, approximately 8 cm deep soil. The seeds were put into the soil on January 5th, and most of them did not grow to to full size. The author estimates that the fully grown individuals were around 50 in number by harvest time, which was February 2nd. The experiments presented later in this thesis were carried out between January 30th and February 2nd.

4.4.1 Parameters of the Experiments

The goal of the experiments was to answer the main hypothesis(1), and hopefully some of the sub-hypotheses (1a, 1b).

How long does an interval need to be in order to measure the fitness of an LED setting?

In order to estimate the viability of the experiment cycle(figure 4.3), this sub-hypothesis seeks to find the time frame in which one can expect meaningful results. Each cycle has two main phases where most of the

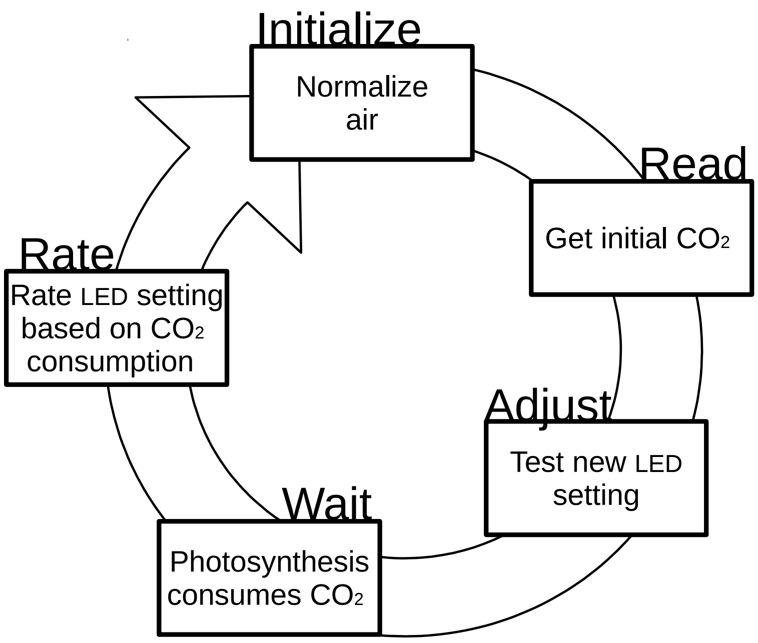


Figure 4.3: The steps that were taken in each experiment cycle.

time is spent; normalization (interval (a, b) in Fig.4.4) and waiting (interval (b, c) in Fig.4.4).

The goal of the *normalization* phase was to provide similar initial conditions for each experiment cycle, so they would be comparable. This was done by opening the dampers and turning on the external fans. The external and internal air circulated until we could se that the CO_2 concentrations had stabilized in both sensors. This was repeated a number of times until it was clear how long it took until we could be sure that the normalization was complete. The LED arrays were turned off during the normalization phase, in order to limit photosynthesis (and consequently speed up normalization).

Next was the *waiting* phase. The phase is commenced by switching the LED arrays to a new setting, and storing the initial CO_2 levels. Then the dampers are closed and external fans switched off. The internal sensor fan keeps going to stir the air around the sensor. Readings are carried out at short intermissions (1 minute). In order to fully answer sub-hypothesis 1b we ran a number of experiments with the same normalization time (determined from the previous observations). The minimal wait time for photosynthesis to reach its peak production would be determined by observing the *stable low* point (illustrated in fig 4.4) at which CO_2 levels bottomed out in all of the experiments.

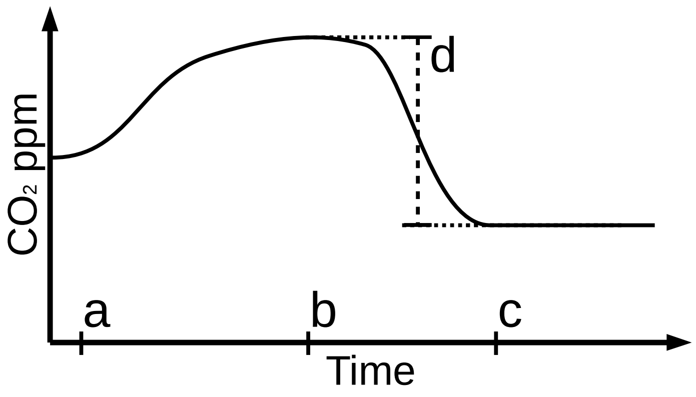


Figure 4.4: An illustration of one experiment cycle: (a) The normalization phase starts. When external and internal air reach equilibrium (b) CO_2 level is recorded, LEDs are adjusted to a new *setting* and enclosure is isolated from outside air. Compare concentration at the stable low point (c) to the one at (b). The difference (d) is the *fitness of the setting*.

Is it possible to estimate photosynthetic action reliably by only measuring CO_2 concentration over time?

The way that we seek to answer this hypothesis is to examine whether the results of these experiments are repeatable. The repeatability will be measured by the similarities in fitness when intervals tesing the same setting are repeated.

An interval will span exactly one measurement cycle, the parameters of which depends on the answer to hypothesis 1b (length of interval). The spread of the data may depend on many external factors such as ambient air composition, lighting, unknowns in plant physiology, to name a few. The key is to determine whether the Telaire 6613 CO_2 sensor measurements can shine a light through all these extra factors so that we may estimate photosynthetic action reliably.

Four Approaches to Estimate the Fitness of a Setting

Once the stable low point of CO_2 has been found (point (c) in fig 4.4), three different approaches of estimation will be compared.

1. Derive the fitness from the simulation described above.
2. Compare stable low concentration to the concentration right after normalization time ends (b in figure 4.4).
3. Compare the stable low point to external CO_2 levels (which may differ from levels at (b))

4. Compare the stable low to a static point, which we'll choose as 400 ppm because that approximates the ambient levels of atmospheric air.

Chapter 5

Data Collection

5.1 Simulation

Setting #	Red	White	Blue	Fitness
a	1	1	1	0.440
b	0.5	0.5	0.5	0.220
c	0.294	0.7	0	0.305
d	0	0.866	0	**0.347**
e	0.09	0.78	0.2	0.332
f	0.50	0.58	0	0.276
g	0.265	0.6	0.7	0.301

Table 5.1: Settings c through g were the results of searches on the simulation. They were hand-picked to have a diverse set to test in the real world. Settings a and b were included as base settings to compare against. Note that setting a violates the energy allowance.

5.2 Enclosure Experiments

5.2.1 Testing Settings Derived from the Simulation

By analyzing the experiments (figure 5.1), the 9 minutes after a 7 minute normalization looked like the earliest time *stable low* could be reached.

The box plots of settings 5.1a and 5.1b indicate that these data had few outliers, which is evident by how close to the edge of the box the median is. The data on settings 5.1d and 5.1f on the other hand have a big spread in their readings. Setting 5.1f has an outlier around 750 ppm, which may indicate that someone was standing close to the enclosure, or peering in. The readings on settings 5.1c, 5.1e and 5.1g are indications that the sensors are relatively stable if undisturbed.

As a whole, it looks like readings are heavily influenced by external conditions - even the internal sensor - but external conditions (e.g. how many students were in the room at the time etc.) weren't noted for any of the measurements.

Even to the naked eye, the biggest difference between two settings looks like setting 5.1a (best) and setting 5.1c (worst), but we'll compare them more thoroughly below.

5.2.2 Comparison of Approaches to Estimation

The performances of each interval of the 7 settings are compared by looking at fitness in 3 distinct ways (figures 5.2, 5.3, 5.4).

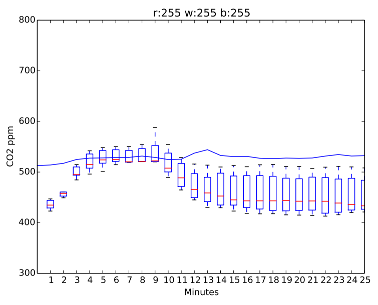
Figure 5.2 shows the fitnesses estimated as the difference between *CO_2 concentration right after normalization* and at the stable low. The fitnesses of most of the methods vary by ~50. Setting (g) is the notable exception to this, and its outliers are also relatively close to the box.

Figure 5.3 shows the fitness estimated as the difference between a *static 400 ppm (atmospheric)* and CO_2 concentration at the stable low. The settings look a bit more equal in this figure compared to the other two, in terms of median, but their differences in variance is exacerbated.

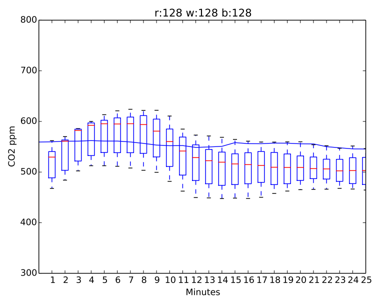
Figure 5.4 shows the fitness estimated as the difference between the *mean value of external CO_2 throughout that interval* and the stable low. This approach seems to have dampened the variance compared to the other approaches. Setting (f) is the only one with a very distant outlier.

Method	Ordering (mean value)						
Simulation (table 5.1)	a	d	e	c	g	f	b
Equilibrium (figure 5.2)	g	a	f	c	b	e	d
400 ppm (figure 5.3)	g	f	a	d	e	c	b
External CO_2 (figure 5.4)	a	g	c	f	b	e	d

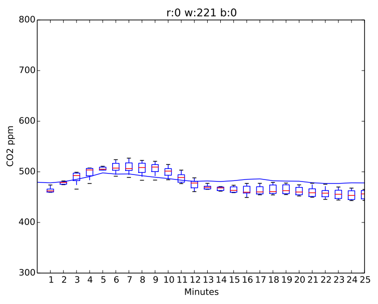
Table 5.2: Settings-wise comparison of the evaluation methods of fitness. Ordering is based on fitness value of the setting when compared using the different methods. Settings are ordered from best to worst (left to right)



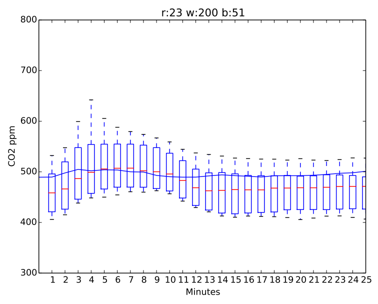
(a) Setting:[255,255,255] Readings from five intervals between 09h and 11h



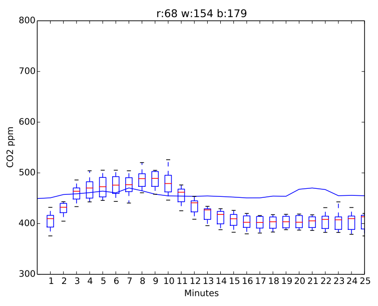
(b) Setting:[128,128,128] Readings from five intervals between 12h and 14h



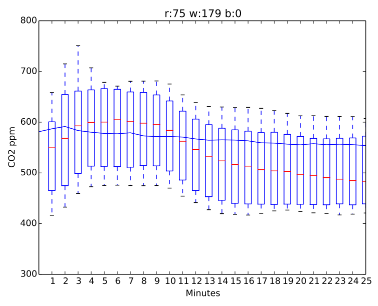
(c) Setting:[0,221,0] Readings from five intervals between 15h and 17h



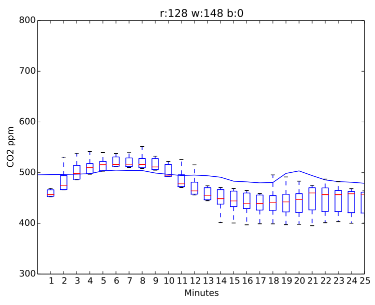
(d) Setting:[23,200,51] Readings from five intervals between 10h and 12h



(e) Setting:[68,154,179] Readings from five intervals between 13h and 15h



(f) Setting:[75,179,0] Readings from five intervals between 16h and 18h



(g) Setting:[128,148,0] Readings from five intervals between 09h and 11h

Figure 5.1: The seven settings from table 5.1 tested in the enclosure. Each setting was tested five times, with readings taken every minute for 25 minutes. Normalization time was 7 minutes. The comparison of intervals is represented by a box spanning the lower and upper quartile (25%-75%) of readings, and a red line at the median. The whiskers show the extremities of the data range. The x-axis shows CO_2 concentration (ppm) between 300 and 800. The y-axis indexes the minutes of the experiment intervals. The blue line is the mean of external CO_2 levels across intervals.

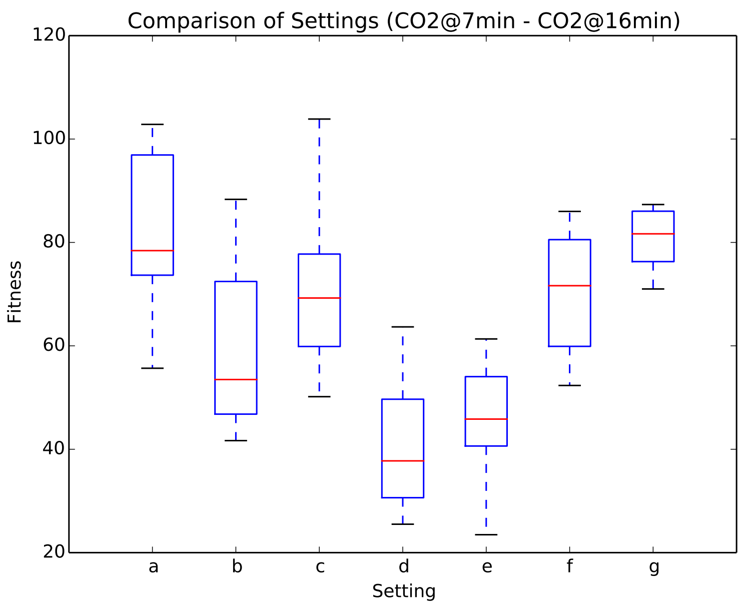


Figure 5.2: Fitness treated as the difference between CO_2 right after normalization (at 7 minutes) and CO_2 at the 16-minute mark.

Plots were generated using the Python module matplotlib.pyplot.

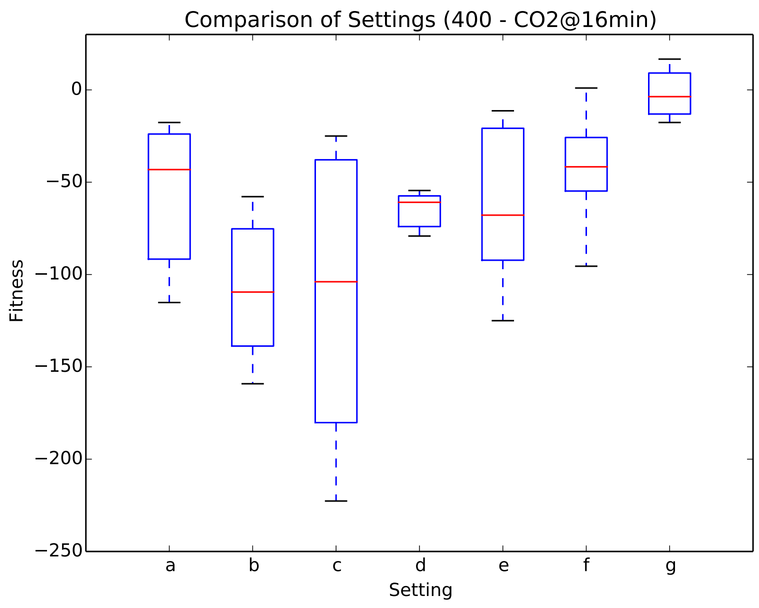


Figure 5.3: Fitness treated as the difference between ambient CO_2 concentration(400 ppm) and CO_2 at the 16-minute mark.

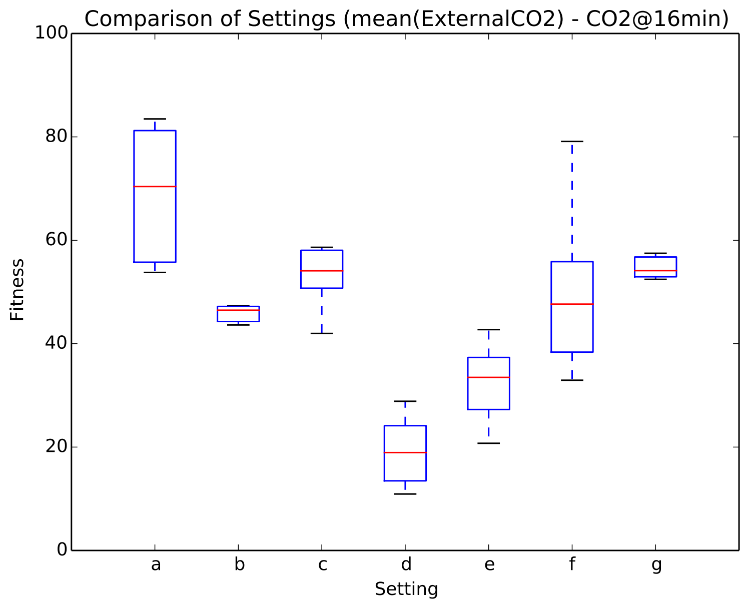


Figure 5.4: Fitness treated as the difference between the mean of external CO_2 concentration during that interval and CO_2 at the 16-minute mark.

Chapter 6

Conclusion

6.1 Discussion

6.1.1 Simulation

White Channel

When trying to investigate if there was any correlation between the fitness derived from simulation and that from reality, the reality gap looked quite chaotic. What *can* be gleaned from this investigation was that the simulation definitely overvalued the white channel because setting d - which was comprised completely out of warm white light - fared very poorly when compared to reality(table 5.2).

Blue Channel

The blue channel also appeared to have been heavily undervalued in the simulation as the setting with the most blue in it (g) had the best fitness overall (even outperforming setting a, which didn't adhere to the energy allowance).

Red Channel

The red channel is difficult to evaluate because so much of it is overshadowed by white's power distribution. Both power distributions(figure 4.2) peak around the same wavelength (660 nm and 645 nm), and white fills the whole spectrum from 400 to 700 nm, while hyper red has a very sharp peak with shoulders around 600 and 700 nm.

Evaluating the Method

In retrospect, it becomes clear that the simulation should have taken into consideration the intensity of light that was output at each waveband from the channels. The power distributions were treated as if the peak of each channel represented equal intensities of light.

The early assumption that the roof intensity of each channel was equal has probably rendered the simulation useless, but the method could still be useful. If the data that the execution is founded on had been collected more wisely, then it's possible that the reality gap could be examined in a useful way.

6.1.2 Enclosure experiments

The readings shown in figure 5.1 show that the readings are affected by external CO_2 concentration to a large degree. The external conditions for 5.1f were around 100 ppm higher than 5.1c, this appears to have a large impact on the variance between intervals.

The some of the settings appear to yield very stable results (figures 5.1e, 5.1c and 5.1g).

The contour of the graphs seems to show that there are distinct differences between some of the settings. It also looks like there's some predictable consistency in determining when the *stable low* CO_2 levels is reached (16 minutes).

Another consistent ocurrence is that it takes about 2 minutes for CO_2 levels start declining after normalization ends and the LEDs are adjusted to the new setting (at the 7-minute mark). This is likely the cause of either sensor response times, or plant response times. Sensor response times are documented as >2 minutes in the datasheet[31], though they have shown almost immediate response when tested for it.

Estimation Approach: Stable low compared to external CO_2 levels

It appears that measure of performance can be gleaned at the stable low point, though the "true fitness" cannot be conclusively determined from these data. Out of the three methods tested in the enclosure, the one that took external CO_2 into consideration (figure 5.4) looks closest to the expected reality[21]: setting (a) (had all channels at 100%) yielded most action, while setting (d) (omitted red and blue channels completely) yielded the lowest action.

Among the settings that adhered to the energy allowance, this method managed to yield the most distinct differences. The two best settings (settings (c) and (g), table 5.2) of them had a fitness of around 50-60, which is decidedly more than setting (d), scoring between 10-30.

Estimation Approach: Stable low compared to CO_2 levels after normalization

The method which used the CO_2 levels at the end of the normalization phase (figure 5.2) ranked the settings very similiarly. Its weakness is that it uses a single data point for its comparison, whereas the previously mentioned method uses the mean of the whole interval for comparison.

Estimation Approach: Stable low compared to atmospheric CO_2 levels

The third method does not seem to work in the context of these experiments. The goal was to measure the photosynthetic action against atmospheric levels of CO_2 to see whether the settings could be ranked that way. The result ends up being nonsensical because all the uncertainty baked into the measurements really shine through when standing by themselves. At least it shows the effect of comparing one measurement to another measurement (considering atmospheric CO_2 is well measured) that isn't exposed to the same set of errors as itself.

6.2 Conclusion

6.2.1 Hypotheses

What sources of disturbance are there in the experiments?

Many potential sources of disturbance were mentioned in the Method and Evaluation chapter, and the experiments did not yield any conclusive information about their influence. The most notable source of disturbance appeared to come from students working in the same room as the enclosure, increasing the external CO_2 levels. The external CO_2 sensor was the only instrument which uncovered the influence of any disturbance, but the enclosure itself was obviously built to isolate the plant themselves from the brunt of exposure.

How long does an interval need to be in order to measure the fitness of an LED setting?

The minimal length of an interval that we could conclude is 16 minutes; 7 minutes for normalization and 9 minutes to reach a stable low. If the 2 minute delayed impact on CO_2 after normalization was due to slow sensors, this means that the wait time for the stable low could be shortened to 7 minutes. Normalization time is clearly very dependant on the design of the enclosure and its mechanical components. With more fans for ventilation, normalization could probably be shortened to a minute, if not less.

Is it possible to estimate photosynthetic action reliably by only measuring CO_2 concentration over time?

Even if the methods employed here couldn't estimate photosynthetic action reliably, they seem to suggest that it's possible with the right equipment.

What is required to be able to tune a 3-channel LED array to optimize photosynthetic action within a given energy allowance?

The framework that was described in this thesis fulfills the requirement that *it should be able to control the spectrum and intensity of light inside the enclosure*. The first half of this hypothesis is answered and fulfilled because the setup is equipped with a pair of 3-channel LED arrays which the framework has full control over.

The second half of the main hypothesis - optimizing photosynthetic action within a given energy allowance - depends on the sub-hypotheses. Because they couldn't be conclusively answered, neither can the main hypothesis.

6.2.2 Final Thoughts

The methods that were applied didn't yield conclusive results, but there did appear to be clear differences between the settings when compared against the mean of external CO_2 across the whole interval.

If the sources of error can be controlled or taken into consideration it is possible that this framework can be able to rank LED settings by the rate at which they stimulate photosynthetic action, but it requires more work to achieve that goal.

6.3 Future Work

The results shown are hinting at the possibility of measuring fitness using just a couple of CO_2 sensors and LED arrays, and the framework was intended as a base for further research.

6.3.1 Improvements on the Framework and Enclosure

Isolation

There is a big potential for improvement in the enclosure, and particularly the lid and the light screens should have a high effort-to-effect ratio. The lid leaves cracks that let air through, and this can be remedied quickly by a little planning and handiwork. The light screens were simply an oversight that was discovered all too late in the project.

LED Arrays

Custom LED arrays with many more channels. It would increase the size of the search space and the complexity of the search as a result, but the potential to fit to the plant's curve would be far greater. Maybe with an initial phase where the lights trace the PAR range, one waveband at a time. If this initial phase yielded good results, it could be possible to use that data to make a better simulation.

Database

The database could have a separate table for LED settings which intervals would point to. This could facilitate the evaluation of LED settings over a long period of time in different conditions. Maybe, with time, the framework database would become a catalog of settings with fitnesses based on a history of experiments.

Plant species could also get its own table so that the database could keep track of a larger system of experiments. Relational databases are very flexible by nature, and a lot of different tweaks can be made to fit the application at hand.

New Applications

The framework is somewhat modular, so it would be interesting to see it applied to more complex problems, such as running multiple experiments in parallel.

6.3.2 Improvements on the Methodology

Simulation

It could be useful to map out the solution space in order to more wisely choose an appropriate optimization technique, and also because it could be interesting to see it visualized.

Infrared and Ultraviolet

Some species may be able to utilize light outside the conventional PAR range, and there could be interesting research to be made with ultraviolet and possibly infrared equipment (though this might interfere with NDIR sensors).

CO_2 Saturation

A CO_2 delivery system could be installed, and it would be interesting to see the effect of reaching the upper limit of the gas as well as light.

6.4 Acknowledgements

Assembling the circuitry and debugging the various interfaces and protocols involved was a lot more time consuming than expected. The instant feedback of programming with physical circuits is incredibly rewarding, and the assembly is itself immersive. The joy of dabbling in new practical skills was a huge motivator when working on this project.

Thanks to Mats Erling Høvin for building the enclosure scaffolding, plexiglass walls and dampers (servos and fans), and for guidance with a wide range of problems from electronics to the scientific method.

Figures were made by the author using LibreOffice Draw

Appendices

Appendix A

Source Code

Full source code and more info in the README at[1]

A.1 Arduino

The code that runs on the microcontroller, with libraries for Telaire 6613, HYT271 and Servo communication. Main code (protocol.ino) implements the communication protocol between Arduino and Raspberry Pi that is described in the thesis.

A.2 Raspberry Pi

Communcation protocol with Arduino. Database interface and schema. Server code to host the presentation of sensor data. Numerous scripts that run specific parameterized experiments in the enclosure.

A.3 Search

Contains the code used to run the optimization algorithms Hill Climber and Simulated Annealing.

[1]https://github.com/bjornarprytz/led-action-framework/tree/master/source

Appendix B

Pictures of the Setup

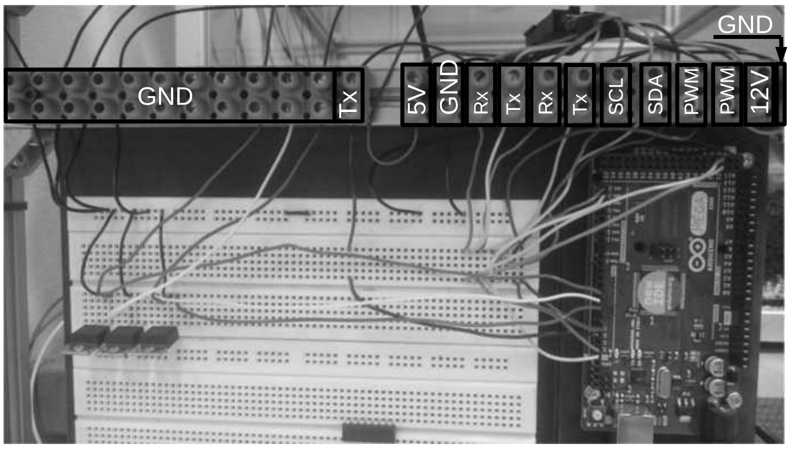


Figure B.1: Microcontroller, breadboard and screw terminals 2 (left) and 3 (right)

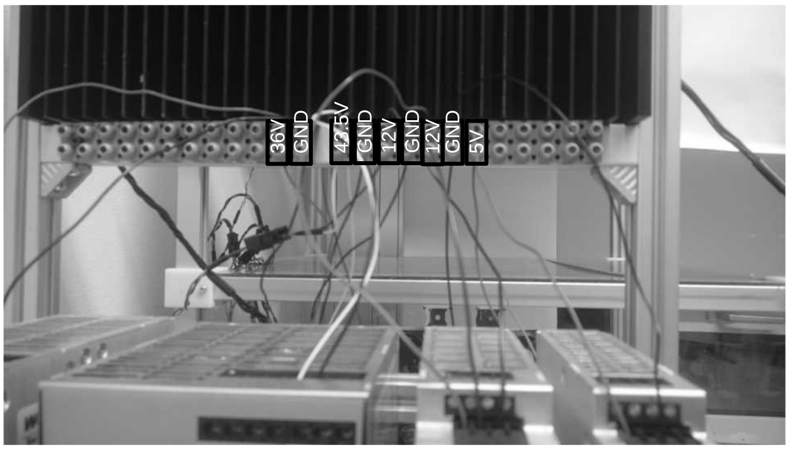


Figure B.2: Powersupplies and screw terminal 1

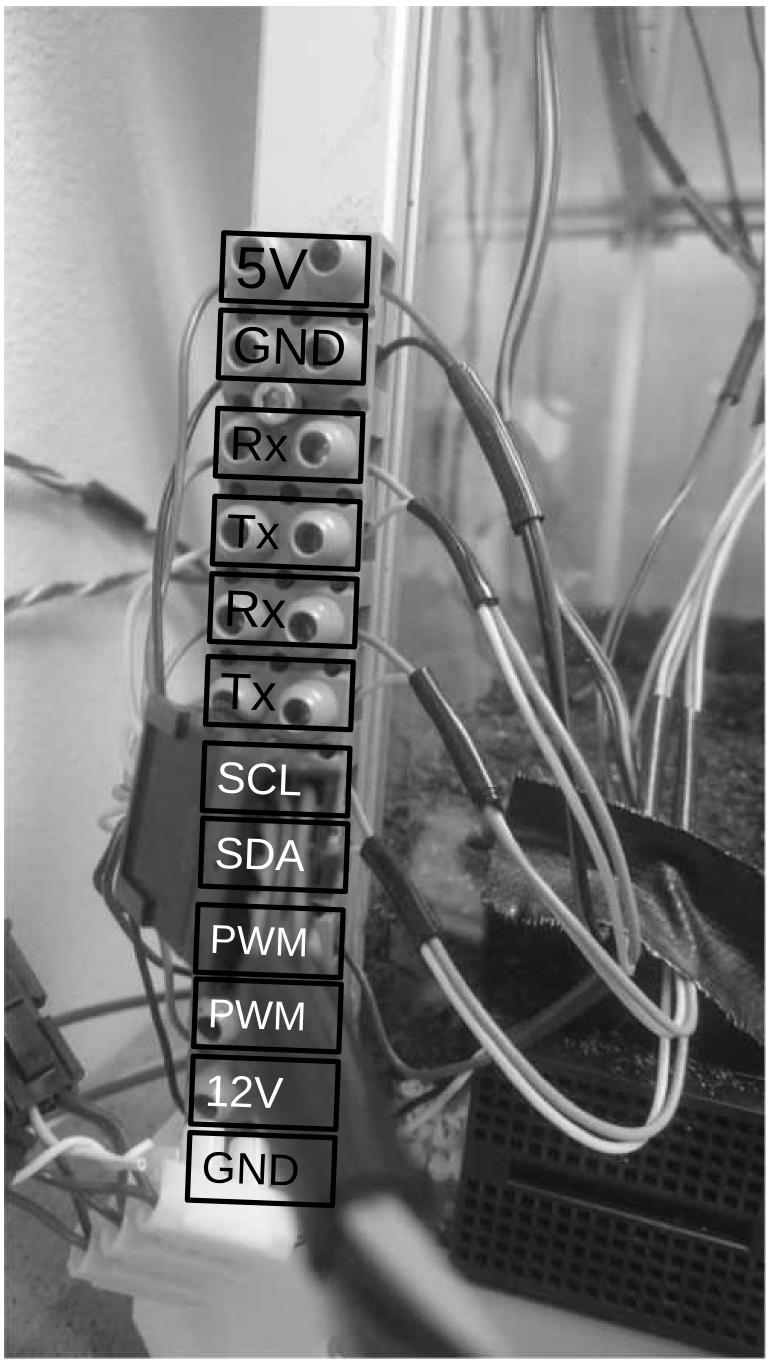


Figure B.3: Screw terminal 4

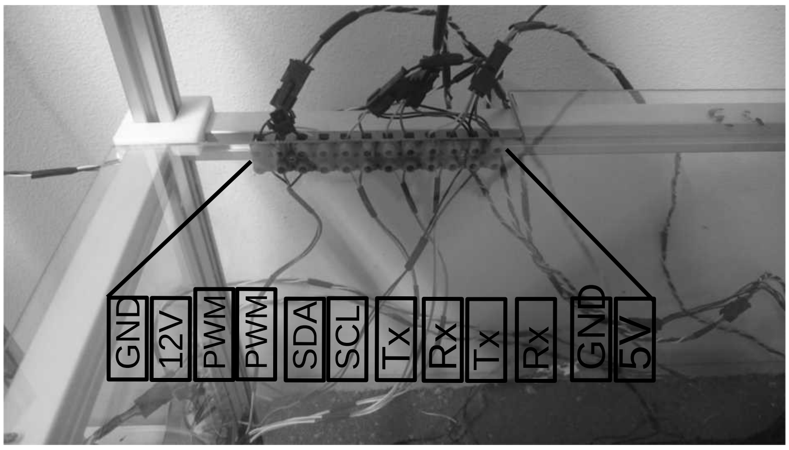


Figure B.4: Screw terminal 5

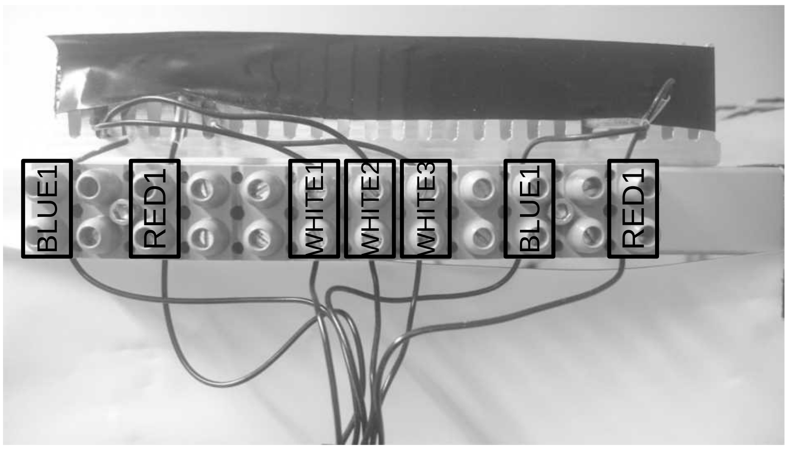


Figure B.5: Overview of terminal 6 (identical to no.7)

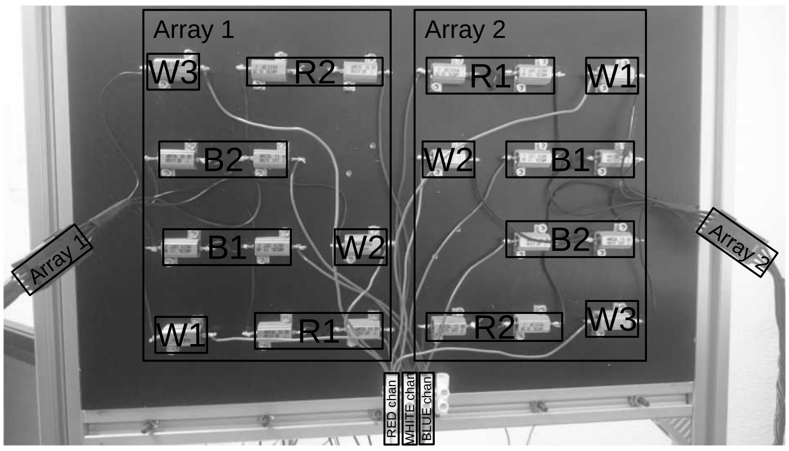


Figure B.6: Overview of the resistors in the LED channels from both arrays.

Bibliography

[1] *12 OSLON ® SSL Petunia*. Version V2. Intelelligent LED Solutions, LEDil, Opto Semiconductors. Jan. 2015.

[2] *12.6. sqlite3 — DB-API 2.0 interface for SQLite databases — Python 3.6.4 documentation*. URL: https://docs.python.org/3/library/sqlite3.html (visited on 28/02/2018).

[3] *Arduino Playground - Participate*. URL: https://playground.arduino.cc/Main/Participate#contribrules (visited on 28/02/2018).

[4] *Arduino - SoftwareSerial*. URL: https://www.arduino.cc/en/Reference/SoftwareSerial (visited on 28/02/2018).

[5] Daniel I. Arnon. 'The Light Reactions of Photosynthesis'. In: *Proceedings of the National Academy of Sciences* 68.11 (1st Nov. 1971), pp. 2883–2892. ISSN: 0027-8424, 1091-6490. URL: http://www.pnas.org/content/68/11/2883 (visited on 28/02/2018).

[6] 'Automatic upgradeable UART circuit arrangement'. 6742057. Neal T. Wingen et al. May 2004.

[7] J. A. Bassham et al. 'The Path of Carbon in Photosynthesis. XXI. The Cyclic Regeneration of Carbon Dioxide Acceptor1'. In: *Journal of the American Chemical Society* 76.7 (1st Apr. 1954), pp. 1760–1770. ISSN: 0002-7863. DOI: 10.1021/ja01636a012. URL: https://doi.org/10.1021/ja01636a012 (visited on 28/02/2018).

[8] *CO 2 measuring instrument testo 535 – Secure monitoring of Indoor Air Quality*. Testo.

[9] *Comparison of single-board computers*. In: *Wikipedia*. Page Version ID: 826134910. 17th Feb. 2018. URL: https://en.wikipedia.org/w/index.php?title=Comparison_of_single-board_computers&oldid=826134910 (visited on 28/02/2018).

[10] *DC axial compact fan*. ebm-papst. June 2016.

[11] *Debian – Det universelle operativsystemet*. URL: https://www.debian.org/ (visited on 28/02/2018).

[12] Gerald F. Deitzer, Rebecca Hayes and Merten Jabben. 'Kinetics and Time Dependence of the Effect of Far Red Light on the Photoperiodic Induction of Flowering in Wintex Barley'. In: *Plant Physiology* 64.6 (1st Dec. 1979), pp. 1015–1021. ISSN: 0032-0889, 1532-2548. DOI: 10.1104/pp.64.6.1015. URL: http://www.plantphysiol.org/content/64/6/1015 (visited on 28/02/2018).

[13] *Frappe Charts*. URL: https : / / frappe . github . io / charts/ (visited on 28/02/2018).

[14] Rajni Govindjee, Govindjee and George Hoch. 'Emerson Enhancement Effect in Chloroplast Reactions'. In: *Plant Physiology* 39.1 (Jan. 1964), pp. 10–14. ISSN: 0032-0889. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC550018/ (visited on 10/10/2017).

[15] Joel Gunter. 'The man who sent his sports car into space'. In: *BBC News* (10th Feb. 2018). URL: http : / / www . bbc . com / news / science-environment-42992143 (visited on 28/02/2018).

[16] *Have you heard about Petunia LED Grow Lights?* URL: https://www.rs-online.com/designspark/have-you-heard-about-petunia-led-grow-lights-2 (visited on 28/02/2018).

[17] Jeongwook Heo et al. 'Growth responses of marigold and salvia bedding plants as affected by monochromic or mixture radiation provided by a Light-Emitting Diode (LED)'. In: *Plant Growth Regulation* 38.3 (1st Nov. 2002), pp. 225–230. ISSN: 0167-6903, 1573-5087. DOI: 10 . 1023 / A : 1021523832488. URL: http : / / link . springer . com / article / 10.1023/A:1021523832488 (visited on 26/01/2017).

[18] *JSON*. URL: https://www.json.org/ (visited on 28/02/2018).

[19] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi. 'Optimization by Simulated Annealing'. In: *Science* 220.4598 (1983), pp. 671–680. ISSN: 0036-8075. URL: http : / / www . jstor . org / stable / 1690046 (visited on 28/02/2018).

[20] *Light pollution depends on the light source CCT (MAGAZINE)*. URL: http://www.ledsmagazine.com/articles/print/volume-12/issue-10/features/street - lights / light - pollution - depends - on - the - light - source - cct . html (visited on 28/02/2018).

[21] Gioia D. Massa et al. 'Plant Productivity in Response to LED Lighting'. In: *HortScience* 43.7 (1st Dec. 2008), pp. 1951–1956. ISSN: 0018-5345, 2327-9834. URL: http://hortsci.ashspublications.org/content/43/7/1951 (visited on 26/01/2017).

[22] K. J. McCree. 'The action spectrum, absorptance and quantum yield of photosynthesis in crop plants'. In: *Agricultural Meteorology* 9 (Supplement C 1st Jan. 1971), pp. 191–216. ISSN: 0002-1571. DOI: 10.1016 / 0002 - 1571(71) 90022 - 7. URL: http : / / www . sciencedirect . com / science/article/pii/0002157171900227 (visited on 10/10/2017).

[23] D. C. Morgan and H. Smith. 'A systematic relationship between phytochrome-controlled development and species habitat, for plants grown in simulated natural radiation'. In: *Planta* 145.3 (1st Jan. 1979), pp. 253–258. ISSN: 0032-0935, 1432-2048. DOI: 10.1007/BF00454449. URL: https://link.springer.com/article/10.1007/BF00454449 (visited on 28/02/2018).

[24] Nathan Nelson and Charles F. Yocum. 'Structure and Function of Photosystems I and Ii'. In: *Annual Review of Plant Biology* 57.1 (2006), pp. 521–565. DOI: 10.1146/annurev.arplant.57.032905.105350. URL: https://doi.org/10.1146/annurev.arplant.57.032905.105350 (visited on 28/02/2018).

[25] O.H.Blaauw. *Natuurtijdschriften*. natuurtijdschriften. Oct. 1970. URL: http://natuurtijdschriften.nl/search?identifier=539723 (visited on 28/02/2018).

[26] *Relational model*. In: *Wikipedia*. Page Version ID: 820381254. 14th Jan. 2018. URL: https://en.wikipedia.org/w/index.php?title=Relational_model&oldid=820381254 (visited on 28/02/2018).

[27] Richard D. Robarts and Tamar Zohary. 'Temperature effects on photosynthetic capacity, respiration, and growth rates of bloomforming cyanobacteria'. In: *New Zealand Journal of Marine and Freshwater Research* 21.3 (1st Sept. 1987), pp. 391–399. ISSN: 0028-8330. DOI: 10.1080/00288330.1987.9516235. URL: https://doi.org/10.1080/00288330.1987.9516235 (visited on 28/02/2018).

[28] A. Schwartz and E. Zeiger. 'Metabolic energy for stomatal opening. Roles of photophosphorylation and oxidative phosphorylation'. In: *Planta* 161.2 (1st May 1984), pp. 129–136. ISSN: 0032-0935, 1432-2048. DOI: 10.1007/BF00395472. URL: https://link.springer.com/article/10.1007/BF00395472 (visited on 28/02/2018).

[29] Emil L. Smith. 'LIMITING FACTORS IN PHOTOSYNTHESIS: LIGHT AND CARBON DIOXIDE'. In: *The Journal of General Physiology* 22.1 (20th Sept. 1938), pp. 21–35. ISSN: 0022-1295. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2213731/ (visited on 14/02/2018).

[30] *SQL*. In: *Wikipedia*. Page Version ID: 827227757. 23rd Feb. 2018. URL: https://en.wikipedia.org/w/index.php?title=SQL&oldid=827227757 (visited on 28/02/2018).

[31] *Telaire 6613 CO 2 Module Small, Compact CO 2 Module Designed to Integrate Into Existing Controls and Equipment*. Telaire. 2011.

[32] *THEORY AND OPERATION OF NDIR SENSORS*. Rae systems. Feb. 2004.

[33] Brian Thomas and H. G. Dickinson. 'Evidence for two photoreceptors controlling growth in de-etiolated seedlings'. In: *Planta* 146.5 (1st Oct. 1979), pp. 545–550. ISSN: 0032-0935, 1432-2048. DOI: 10.1007/BF00388830. URL: https://link.springer.com/article/10.1007/BF00388830 (visited on 28/02/2018).

[34] J. Zosel et al. 'The measurement of dissolved and gaseous carbon dioxide concentration'. In: *Measurement Science and Technology* 22.7 (2011), p. 072001. ISSN: 0957-0233. DOI: 10.1088/0957-0233/22/7/072001. URL: http://stacks.iop.org/0957-0233/22/i=7/a=072001 (visited on 23/02/2018).