UMassAmherst
The Commonwealth's Flagship Campus

GrowingGreen MDR

Team Members: Austin Hiller, Nate Lemons, Matthew Sargeant, Jason Trainor

Advisor: Professor Kundu

12.6.2019

Who We Are

Austin Hiller CSE

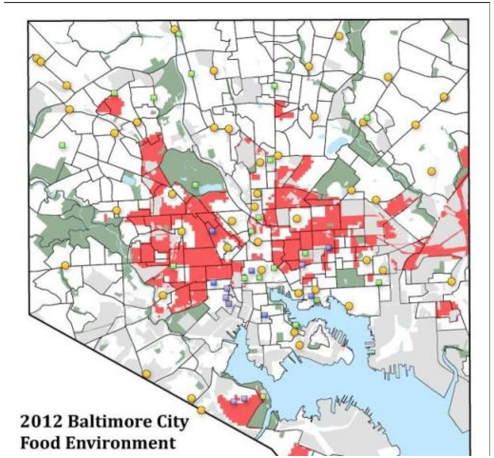
Nate Lemons CSE

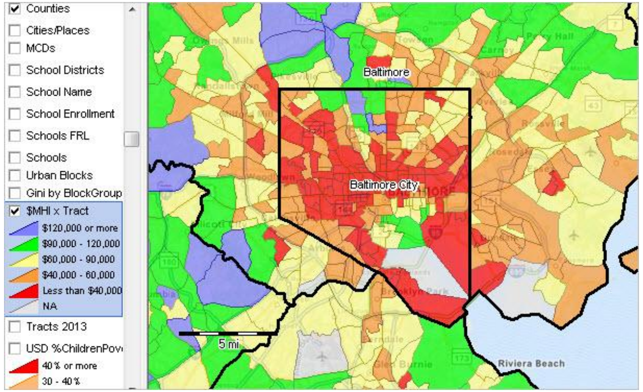
Matthew Sargeant EE

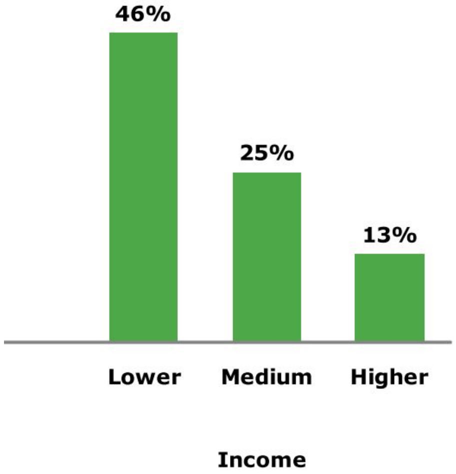
Jason Trainor CSE

Problem Statement

- The Grocery Gap
  - Low income zip codes have 25% fewer supermarkets
  - Rural and urban communities affected
- Proximity to a supermarket is correlated to healthy diet habits
- Low-income neighborhoods have half as many supermarkets as the wealthiest neighborhoods and four times as many smaller grocery stores



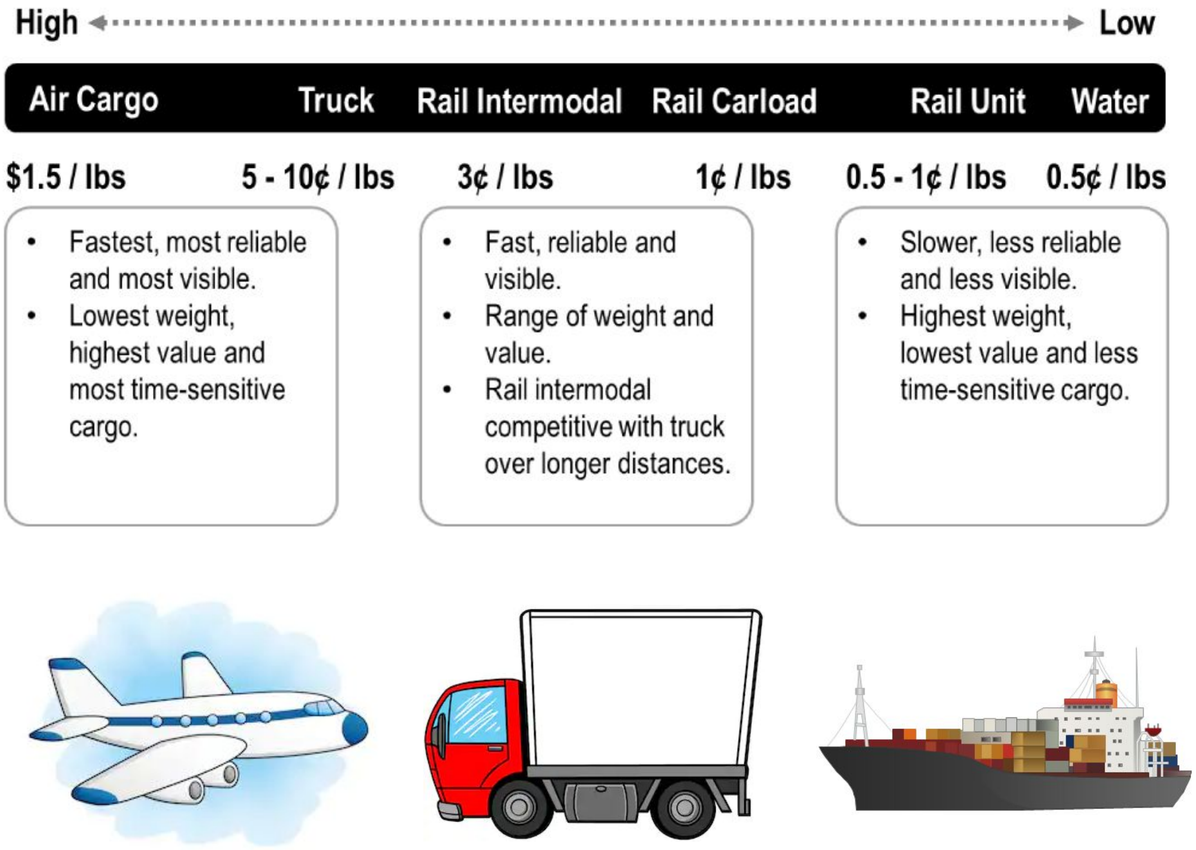




Share of Baltimore neighborhood grocery stores with low availability to healthy food, by income

Problem Statement

- Current food system is very taxing on environment due to transportation
- 10 Kcal of energy from fossil fuel per 1 Kcal of energy from food
- Transport leads to lesser quality, more chemical influence, and higher cost



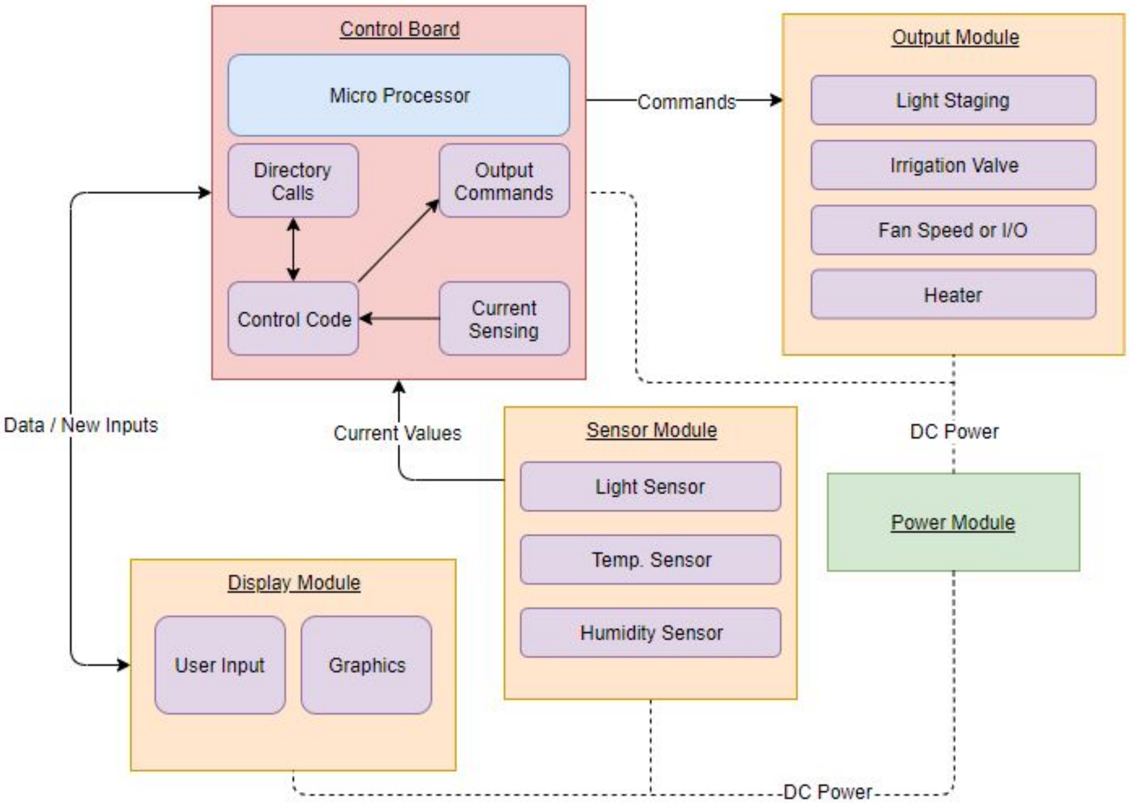
Vision Statement

The GrowingGreen system is a fully automated, energy efficient, in-house growsite with focus on supplying the grower edible vegetation with minimal effort. Our goal is to increase the availability and desire of home growing by simplifying the process through the automation of manual processes, lessening of power consumption, and use of a user console with alerts to keep growers engaged and on schedule. By growing in-house, users will decrease their environmental impact by reducing their carbon footprint and pesticidal use on plants.

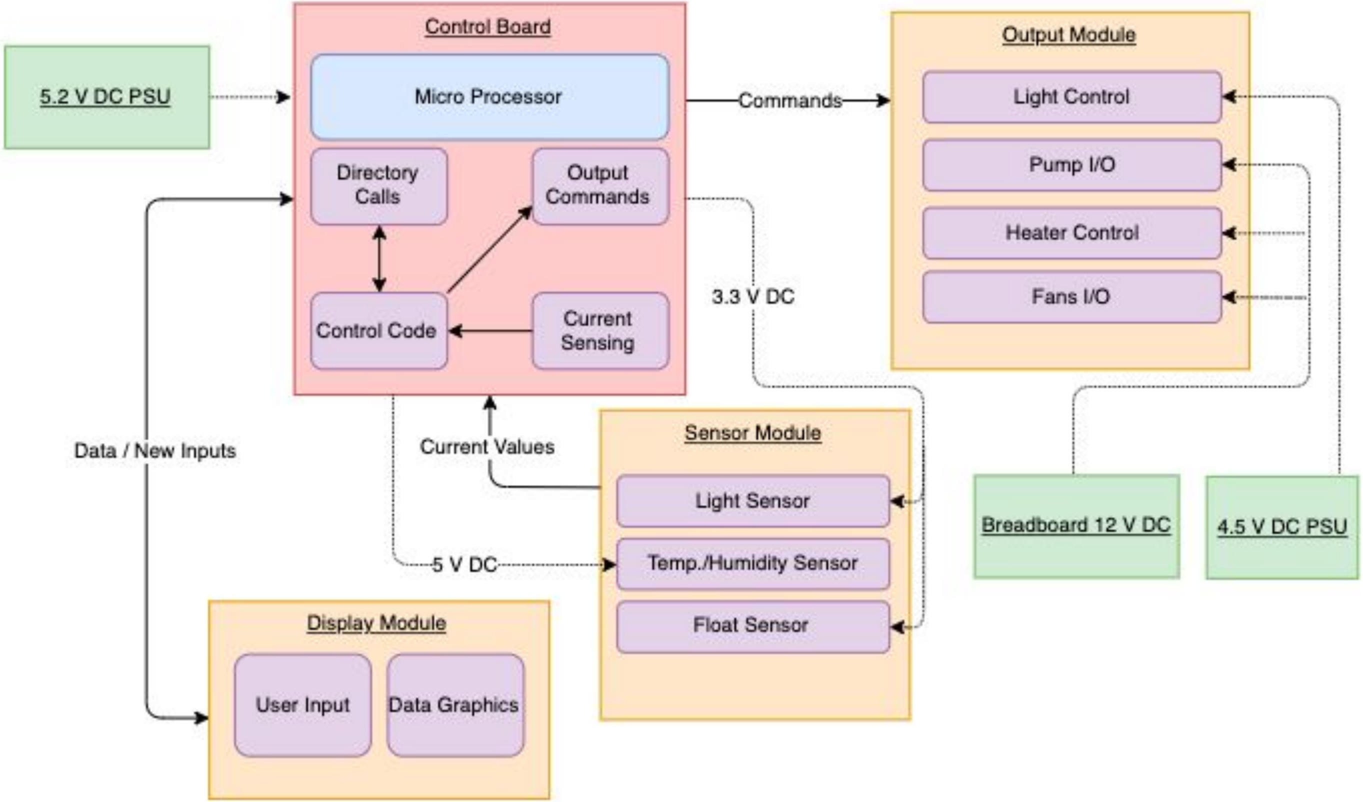
System Specifications

- Reduce power consumption by 3 times the standard
- Produce 24 ounces of product per cycle
- Simplify process of growing so even engineers can grow plants
- Automation through feedback control:
  - Lights
  - Temperature/Humidity
  - Irrigation
- Functional year round
- Data available to user at all times
- Must fit against typical window frame

PDR Block Diagram

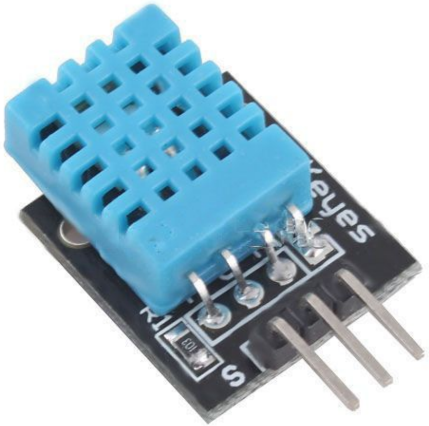


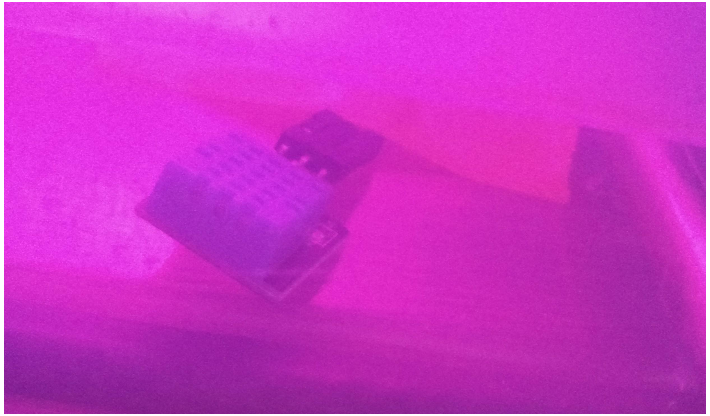
Updated Block Diagram



Subsystem 1: Temp./ Humidity Sensor

- Temperature and Humidity will be measured with a single sensor.
  Keyes KY-015 DHT11
  - 20-90% RH range
  - 3.3-5 V DC Supply



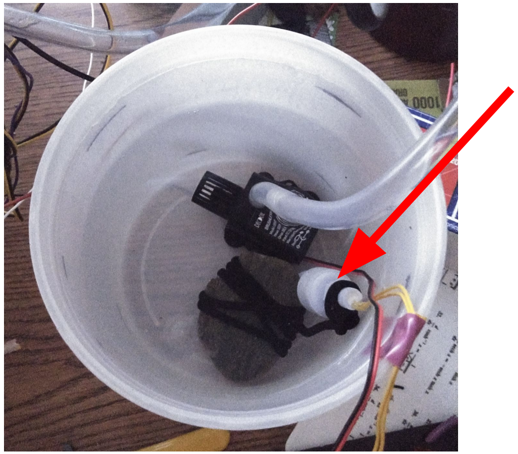


Subsystem 1: Light Sensor

- Insolation sensor will be an LDR.

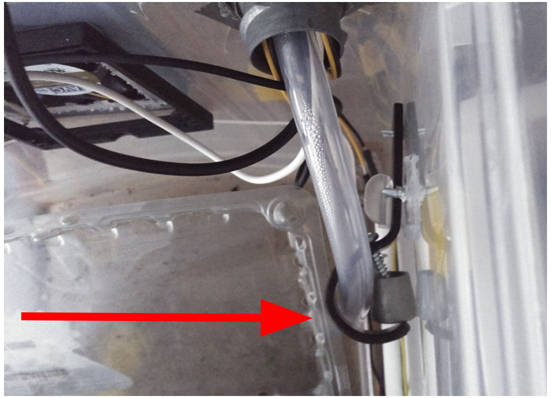
Subsystem 1: Float Sensor

- Anndason PP Float Switch
    - User is notified reservoir is “empty” when button is pressed
        - Pump is turned off



Subsystem 2 Irrigation

- Tray-in-tray system
  - Water supplied from reservoir containing:
    - 12V Decdeal Submersible Water Pump
    - Anndason PP Float Switch
  - Watering cycle is executed each morning
    - Pumps ½ cup of water into bottom tray

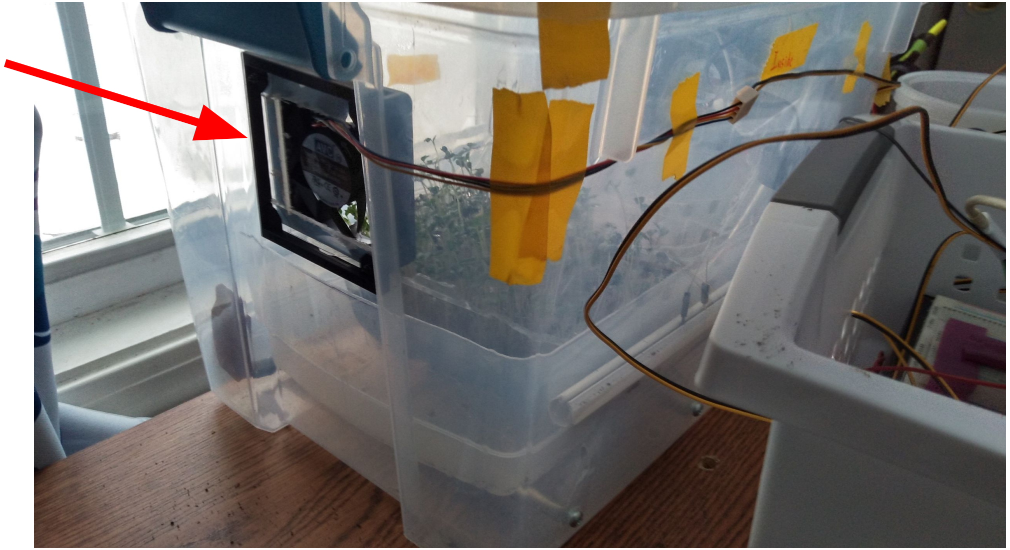


Top watering trial

Bottom watering trial

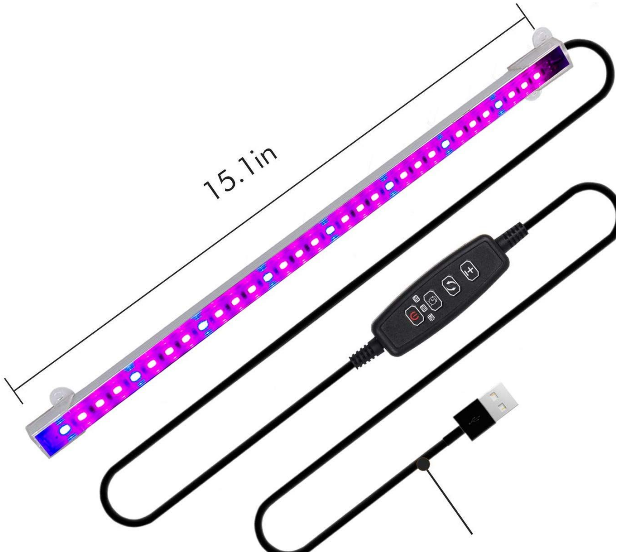
Subsystem 2: Air Flow

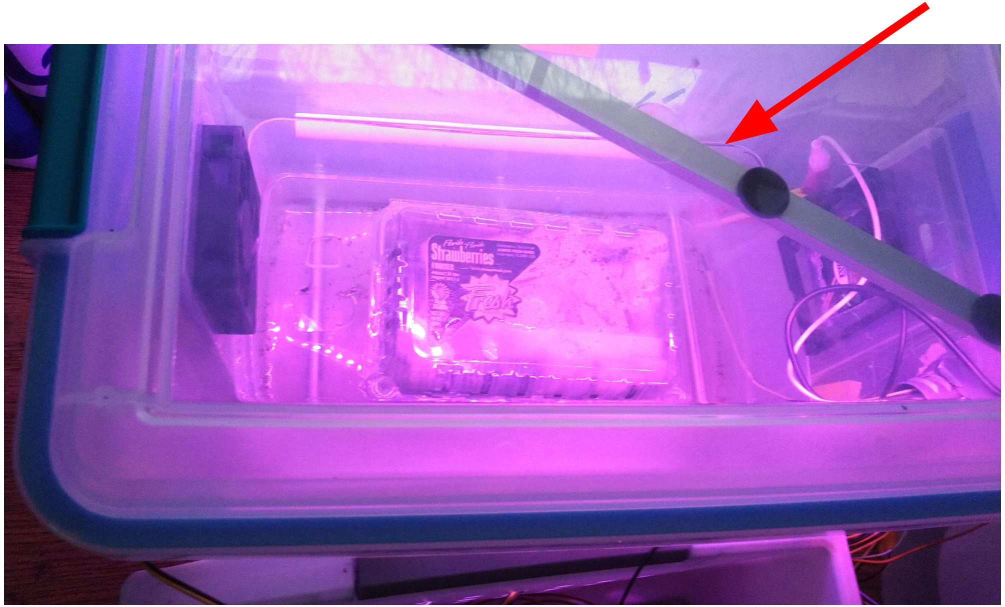
- Two 12V fans on either side of enclosure
  - Fans work with heating coils & temp./humidity sensor
    - If temp./humidity threshold is exceeded → fans turned on to circulate air



Subsystem 2 LED

- Sondiko LED Grow Light Strip
  - Using red and blue light spectrums to supplement direct sunlight
  - Lights supplement lack of direct sunlight to ensure 10 hour light cycle





Subsystem 2: Heating coil

- Controlled using 12V DC Freeze Stop Heating Cable w/5 Watts/ft output
- ~16 in = 6.25 W output
- Self-regulated to 85 F to prevent burning out
- Capable of operating with AC & DC supplies
  - Currently powered by 12V DC supply

Control Code

```
def check_light (light_sensor):
    count = 0
    #Output on the pin for
    GPIO.setup(light_sensor, GPIO.OUT)
    GPIO.output(light_sensor, GPIO.LOW)
    time.sleep(.1)
    #Change the pin back to input
    GPIO.setup(light_sensor, GPIO.IN)
    #Count until the pin goes high
    while (GPIO.input(light_sensor) == GPIO.LOW):
        count += 1
        if count >= 15000:  #its dark, stop counting
            break
    return count
```

```
#fatches data from the csv at specific locations
def get_ideal(row, column):
    with open('/home/pi/GrowingGreen/plants.csv', mode='r') as csvfile:
        data = list(csv.reader(csvfile))
        value = data[row][column]
        return value
#get ideals
ideal_humidity = get_ideal(plant_num,humidity_ideal)
ideal_temp = get_ideal(plant_num,temp_ideal)
```

```
#control an output by light
def control_by_light (light_now):
        if light_now <= 1000: #get_ideal(plant_num,light_ideal):
            #if we use a switch network for power to lights
            print("stage 1 - off")  #the lights are off
            turn_off(light_1)
            turn_off(light_2)
            turn_off(light_3)
            logger.info("LIGHTS OFF")
        if 1000 < light_now <= 5000:
            print("stage 2 - 2.5V on")  #the lights are on their lowest setting
            turn_on(light_1)
            turn_on(light_3)
            turn_off(light_2)
            logger.info("LIGHT SETTING LOW")
        if 5000 < light_now <= 10000:
            print("stage 3 - 3.5V on")  #the lights are on their middle setting
            turn_on(light_1)
            turn_on(light_2)
            turn_off(light_3)
            turn_on(light_high)
            logger.info("LIGHTS SETTING MEDIUM")
        elif 10000 < light_now:
            print("stage 4 - 4.5v")     #the lights are on their highest setting
            turn_on(light_1)
            turn_off(light_2)
            turn_off(light_3)
            logger.info("LIGHTS SETTING HIGH")
```

Control Code

```
#set gpio output pins to variables
GPIO.setup(light_1, GPIO.OUT)
GPIO.setup(light_2, GPIO.OUT)
GPIO.setup(light_3, GPIO.OUT)
GPIO.setup(fan, GPIO.OUT)
GPIO.setup(fan2, GPIO.OUT)
GPIO.setup(heater, GPIO.OUT)
```

```
#output variables
light_1 = 19
light_2 = 26
light_3 = 25
fan = 17
fan2 = 13
heater = 22
```

```
#input variables
temp_humidity = 21
ls_window = 20
ls_inside = 16
light_sensor = 20
```

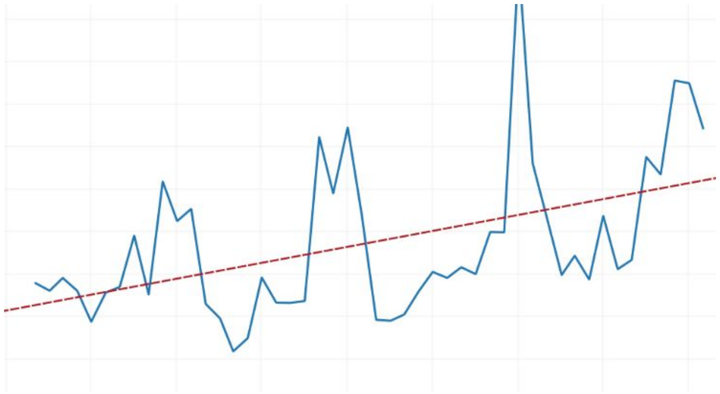
```
#return humidity
def check_humidity ():
    humidity, temperature = Adafruit_DHT.read_retry(11, temp_humidity)
    return humidity

def anomolycheck_hum(humidity_now):
    with open('/home/pi/GrowingGreen/logs/data.log', 'rb') as f:
        f.seek(-2, os.SEEK_END)
        while f.read(1) != b'\n':
            f.seek(-2, os.SEEK_CUR)
        lastline = f.readline().decode()    # Get just the last line
        #including the 11 spaces for what we found to have it grab the starting spot of the sub
        intStart = lastline.find("humidity':") + 11
        #print(intStart) #testing
        sub = lastline[intStart:] #create sub to be the remander of the string
        intEnd = sub.find("}") #get ending marker for the string we want
        finalvalue = sub[0:intEnd] # Finally, grab the value, using
        print finalvalue
        while float(humidity_now) > (float(finalvalue)+15) or float(humidity_now) < (float(finalvalue)-15):
            humidity_now = check_humidity()
            print("anomoly detected - checking again")
        return humidity_now
```

```
#control an output by humidity and temp
def control_temp_humidity (humidity_now, temp_now):
        #humidity > ideal and temp > ideal
        if (humidity_now > float(ideal_humidity) and temp_now > float(ideal_temp)):
            print('option 1')
            turn_on(fan)
            turn_on(fan2)
            print('fan on')
            turn_off(heater)
            deviceloc.info("HEATER OFF")
            print('heater off')
            deviceloc.info("OPTION 1")
            deviceloc.info("FANS ON")
        # humidty <= ideal and temp <= ideal
        if (humidity_now <= float(ideal_humidity) and temp_now <= float(ideal_temp)):
            print('option 2')
            turn_off(fan)
            turn_off(fan2)
            print('fan off')
            #heater_me()
            turn_on(heater)
            deviceloc.info("HEATER ON")
            print('heater on')
            deviceloc.info("OPTION 2")
            deviceloc.info("FANS OFF")
        # humidty > ideal and temp <= ideal
        if (humidity_now > float(ideal_humidity) and temp_now <= float(ideal_temp)):
            print('option 3')
            turn_off(fan)
            turn_off(fan2)
            print('fan off')
            #heater_me()
            turn_on(heater)
            deviceloc.info("HEATER ON")
            print('heater on')
            deviceloc.info("OPTION 3")
            deviceloc.info("FANS OFF")
        # humidty <= ideal and temp <= ideal
        if (humidity_now <= float(ideal_humidity) and temp_now > float(ideal_temp)):
            print('option 4')
            turn_on(fan)
            turn_on(fan2)
            print('fan on')
            turn_off(heater)
            deviceloc.info("HEATER OFF")
            print('heater off')
            deviceloc.info("OPTION 4")
            deviceloc.info("FANS ON")
```

Subsystem 3: Console Interface

- Input: User chooses specific plant to set up variable conditions in greenhouse
- Display: Most recent conditions and trend lines of the greenhouse conditions
- Output: Alerting feature when water reservoir is low or reporting erroneous data
- MDR prototype has simple interface that will output most recent conditions of greenhouse
- MDR prototype has plotting console on our computers for processing power and testing



Logging

- 3 Log Files
  - data_read.log
  - data.log
  - device.log

```
2019-12-03 17:30:14,905 - DEVICE - DATA LOGGED
2019-12-03 17:40:02,108 - DEVICE - LIGHT SETTING LOW
2019-12-03 17:40:15,075 - DEVICE - HEATER ON
2019-12-03 17:40:15,076 - DEVICE - OPTION 3
2019-12-03 17:40:15,077 - DEVICE - FANS OFF
2019-12-03 17:40:15,085 - DEVICE - DATA LOGGED
2019-12-03 17:50:01,905 - DEVICE - LIGHT SETTING LOW
2019-12-03 17:50:30,110 - DEVICE - HEATER ON
2019-12-03 17:50:30,112 - DEVICE - OPTION 3
2019-12-03 17:50:30,113 - DEVICE - FANS OFF
2019-12-03 17:50:30,118 - DEVICE - DATA LOGGED
2019-12-03 19:00:05,020 - DEVICE - HEATER ON
2019-12-03 19:00:05,022 - DEVICE - OPTION 3
2019-12-03 19:00:05,023 - DEVICE - FANS OFF
2019-12-03 19:00:05,026 - DEVICE - DATA LOGGED
```

```
2019-12-04 06:30:29,280 - DATA - {'ohm_inside': 3729664, 'ohm_window': 2623315, 'temp': 66.2, 'humidity': 81.0}
2019-12-04 06:40:11,782 - DATA - {'ohm_inside': 485752, 'ohm_window': 320864, 'temp': 66.2, 'humidity': 80.0}
2019-12-04 06:50:04,681 - DATA - {'ohm_inside': 85701, 'ohm_window': 49664, 'temp': 66.2, 'humidity': 80.0}
2019-12-04 07:00:06,672 - DATA - {'ohm_inside': 21606, 'ohm_window': 12343, 'temp': 66.2, 'humidity': 80.0}
2019-12-04 07:10:06,561 - DATA - {'ohm_inside': 8551, 'ohm_window': 4711, 'temp': 66.2, 'humidity': 80.0}
2019-12-04 07:20:09,127 - DATA - {'ohm_inside': 4461, 'ohm_window': 1907, 'temp': 66.2, 'humidity': 80.0}
2019-12-04 07:30:09,525 - DATA - {'ohm_inside': 3259, 'ohm_window': 1796, 'temp': 66.2, 'humidity': 79.0}
2019-12-04 07:40:07,350 - DATA - {'ohm_inside': 2287, 'ohm_window': 1268, 'temp': 66.2, 'humidity': 79.0}
2019-12-04 07:50:04,666 - DATA - {'ohm_inside': 1520, 'ohm_window': 870, 'temp': 66.2, 'humidity': 80.0}
2019-12-04 08:00:04,561 - DATA - {'ohm_inside': 1085, 'ohm_window': 619, 'temp': 66.2, 'humidity': 78.0}
2019-12-04 08:10:04,523 - DATA - {'ohm_inside': 1018, 'ohm_window': 552, 'temp': 66.2, 'humidity': 77.0}
2019-12-04 08:20:03,979 - DATA - {'ohm_inside': 1477, 'ohm_window': 842, 'temp': 66.2, 'humidity': 78.0}
2019-12-04 08:30:06,886 - DATA - {'ohm_inside': 1154, 'ohm_window': 628, 'temp': 66.2, 'humidity': 80.0}
2019-12-04 08:40:04,260 - DATA - {'ohm_inside': 652, 'ohm_window': 400, 'temp': 66.2, 'humidity': 80.0}
2019-12-04 08:50:04,589 - DATA - {'ohm_inside': 625, 'ohm_window': 369, 'temp': 66.2, 'humidity': 79.0}
```

The System is Working!

Temperature threshold = 70 degrees. Light Threshold 1000 ohms

```
2019-12-04 08:50:04,589 - DATA - {'ohm_inside': 625, 'ohm_window': 369, 'temp': 66.2, 'humidity': 79.0}
```

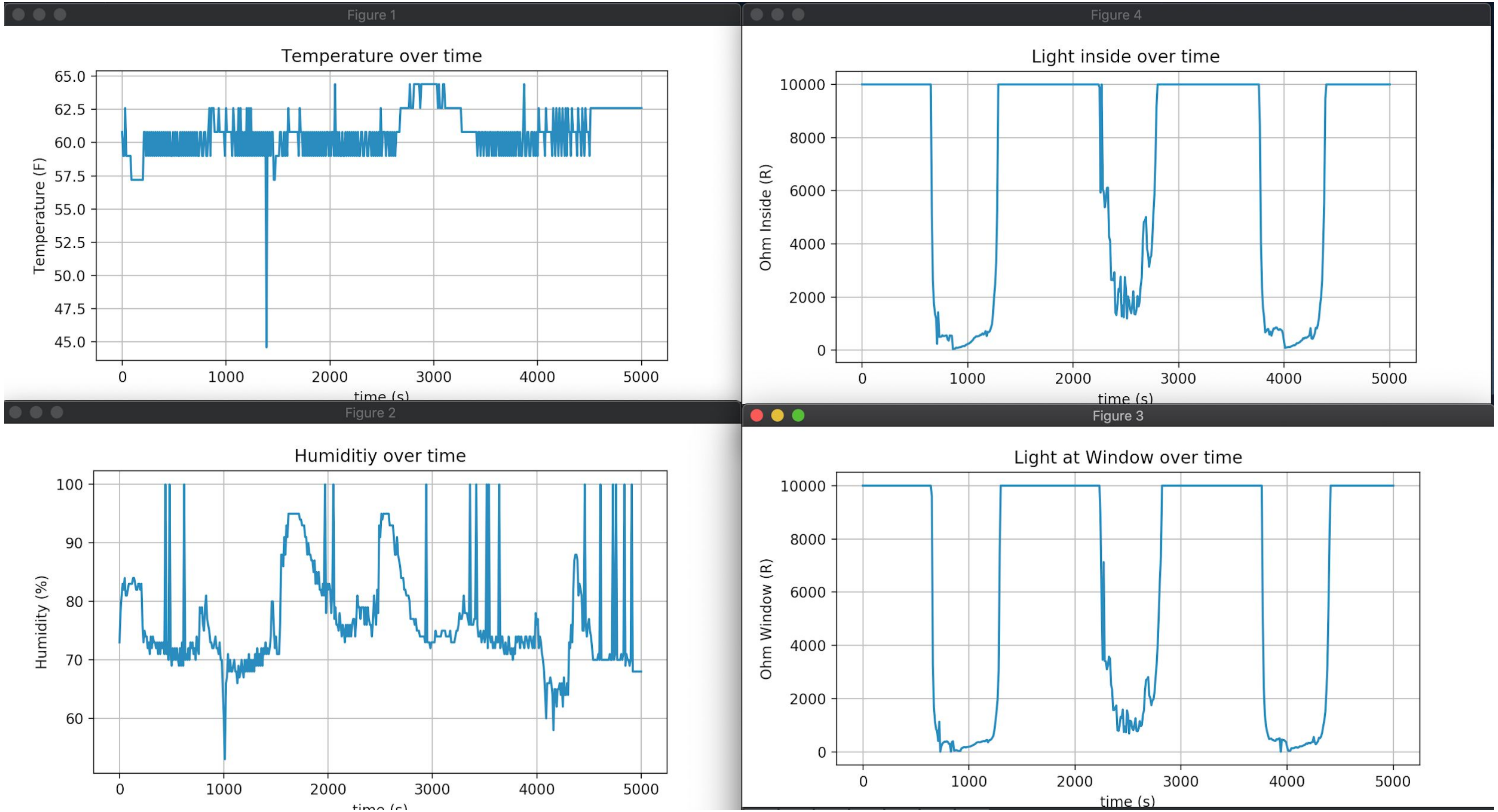
```
2019-12-04 08:50:04,586 - DEVICE - HEATER ON
2019-12-04 08:50:04,587 - DEVICE - OPTION 3
2019-12-04 08:50:04,588 - DEVICE - FANS OFF
2019-12-04 08:50:04,591 - DEVICE - DATA LOGGED
```

Temperature threshold = 60 degrees. Light Threshold 1000-5000 ohms with window taking precedence

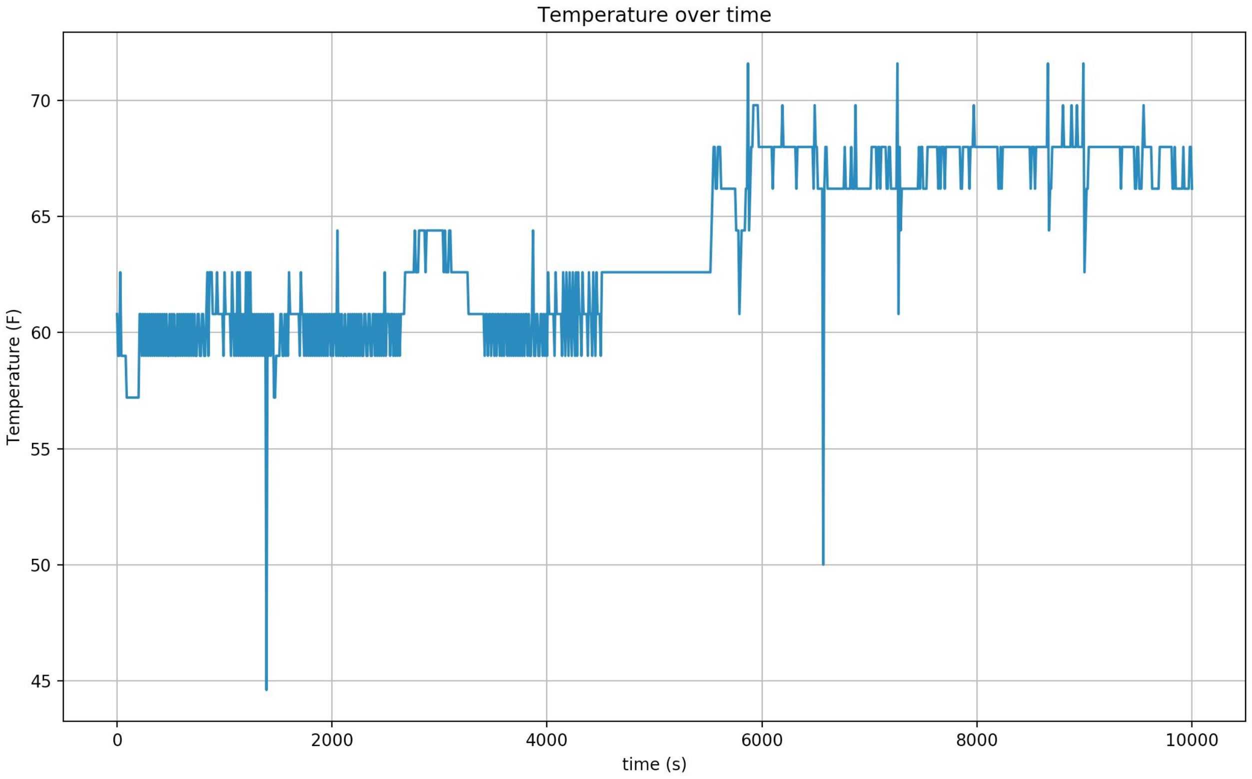
```
2019-11-23 16:10:04,285 - DATA - {'ohm_inside': 5242, 'ohm_window': 3063, 'temp': 60.8, 'humidity': 70.0}
```

```
2019-11-23 16:10:01,501 - DEVICE - LIGHT SETTING LOW
2019-11-23 16:10:04,283 - DEVICE - OPTION 1
2019-11-23 16:10:04,284 - DEVICE - FANS ON
2019-11-23 16:10:04,288 - DEVICE - DATA LOGGED
```

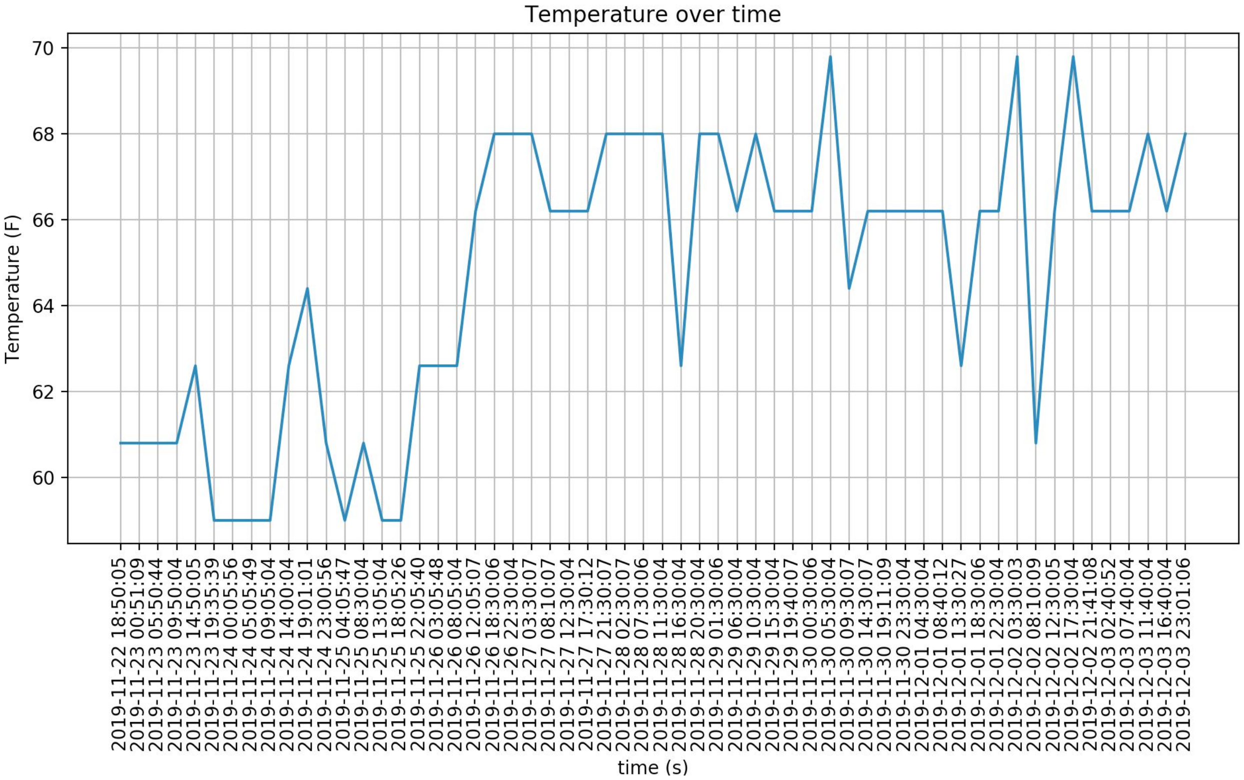
Plotting with real Data (3.5 days)



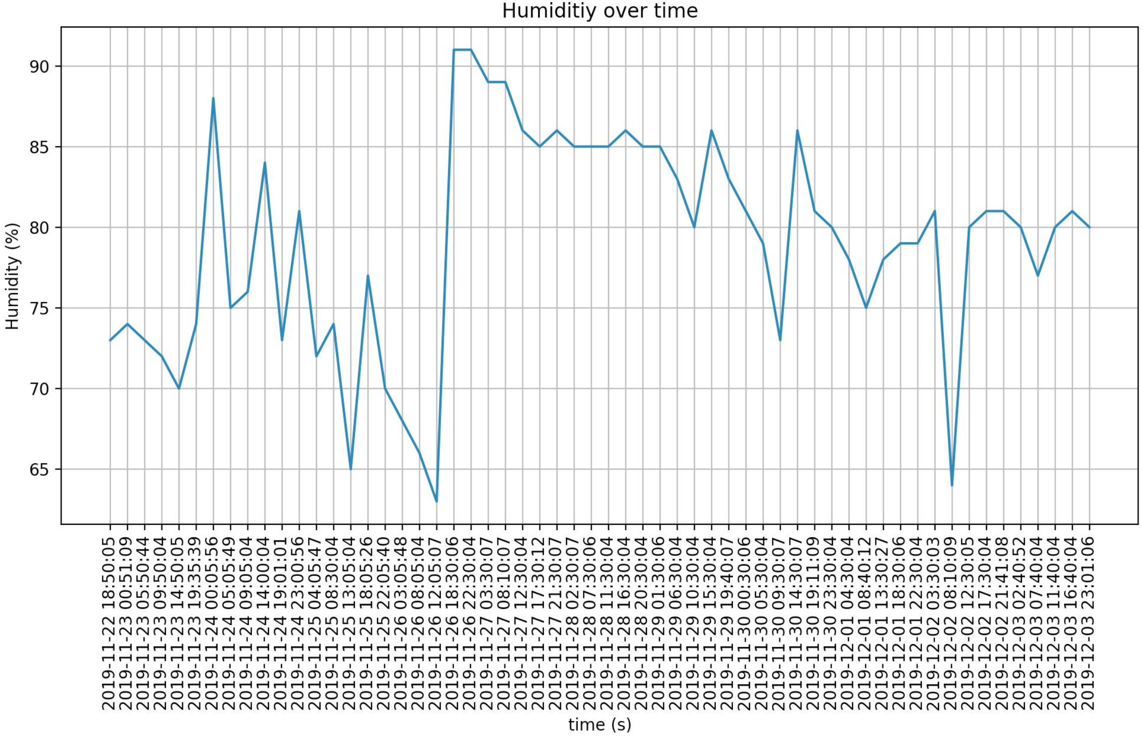
Plotting with real Data: Temperature (1 Week)



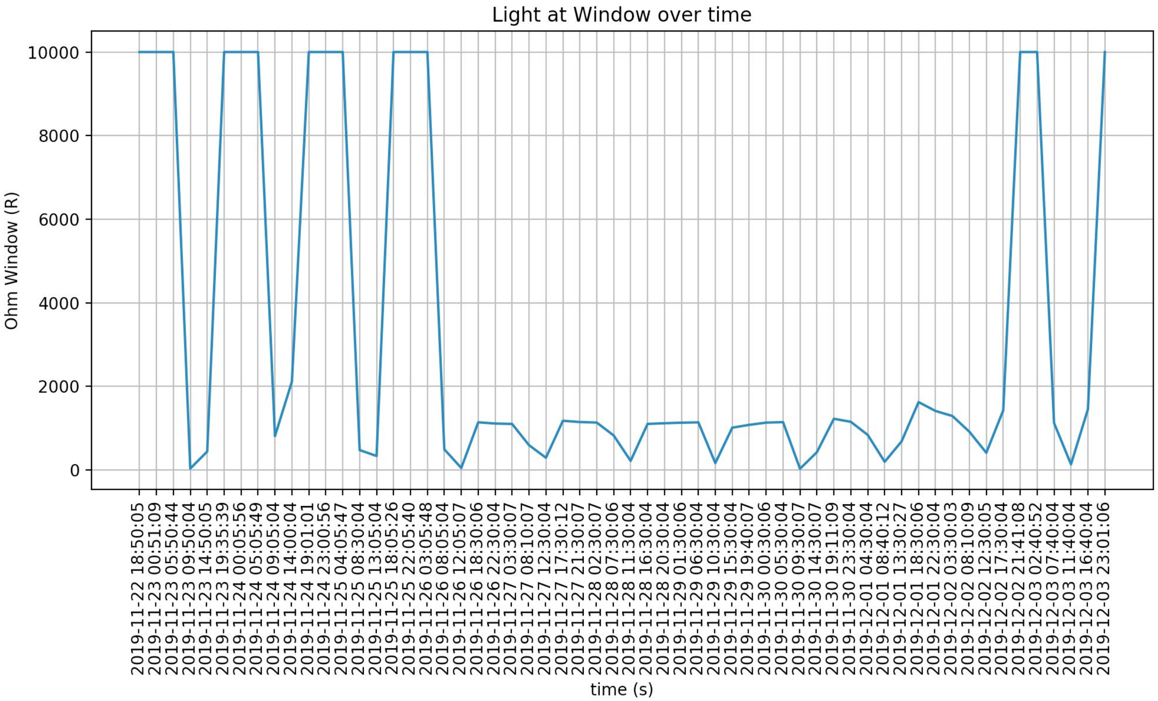
Temp 12 days


Temperature over time
Temperature (F)
70
68
66
64
62
60
time (s)
2019-11-22 18:50:05
2019-11-23 00:51:09
2019-11-23 05:50:44
2019-11-23 09:50:04
2019-11-23 14:50:05
2019-11-23 19:35:39
2019-11-24 00:05:56
2019-11-24 05:05:49
2019-11-24 09:05:04
2019-11-24 14:00:04
2019-11-24 19:01:01
2019-11-24 23:00:56
2019-11-25 04:05:47
2019-11-25 08:30:04
2019-11-25 13:05:04
2019-11-25 18:05:26
2019-11-25 22:05:40
2019-11-26 03:05:48
2019-11-26 08:05:04
2019-11-26 12:05:07
2019-11-26 18:30:06
2019-11-26 22:30:04
2019-11-27 03:30:07
2019-11-27 08:10:07
2019-11-27 12:30:04
2019-11-27 17:30:12
2019-11-27 21:30:07
2019-11-28 02:30:07
2019-11-28 07:30:06
2019-11-28 11:30:04
2019-11-28 16:30:04
2019-11-28 20:30:04
2019-11-29 01:30:06
2019-11-29 06:30:04
2019-11-29 10:30:04
2019-11-29 15:30:04
2019-11-29 19:40:07
2019-11-30 00:30:06
2019-11-30 05:30:04
2019-11-30 09:30:07
2019-11-30 14:30:07
2019-11-30 19:11:09
2019-11-30 23:30:04
2019-12-01 04:30:04
2019-12-01 08:40:12
2019-12-01 13:30:27
2019-12-01 18:30:06
2019-12-01 22:30:04
2019-12-02 03:30:03
2019-12-02 08:10:09
2019-12-02 12:30:05
2019-12-02 17:30:04
2019-12-02 21:41:08
2019-12-03 02:40:52
2019-12-03 07:40:04
2019-12-03 11:40:04
2019-12-03 16:40:04
2019-12-03 23:01:06

Humidity 12 days

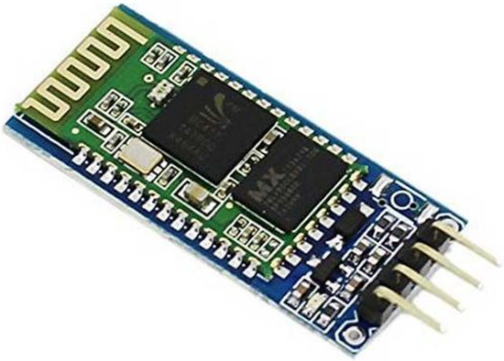


Light at Window 12 days



Printed Circuit Board

- Designed to allocate space for all I/O components; i.e. sensors
  - All sensor will have wired connections
- 8 bit Microcontroller for computations -Update
- Slow clock cycle
- Bluetooth Connection for console -Update



Problems We Faced for MDR

- Power distribution
  - Variable light supply
    - simplified initial approach
- Sensors detecting anomalies
  - Fixed using code presented earlier
- Controlling Temperature
  - Very low R-value prototype

Challenges Moving Forward

- Providing uniform light to prevent reaching
- Providing discrete levels of light
- Maintaining environment all year with low R-value structure
- Variable irrigation needs
- Maintaining plant health
- Integrating onto PCB
- Data Storage

MDR Deliverables Update

- Prototyped enclosure with working sensors and output components ✓
- Control and sensor feedback controlled through dev board ✓
- Automation of manual processes (dev board) ✓
- Successful grow cycle ✓
- Data available for manipulation ✓

Austin Hiller

Console Interface from logs

- Simple program that prints most recent conditions of greenhouse
- Will integrate to an application

Logging

- Logs stored on Pi
- Log creation fluid as data is recorded and devices are triggered
- Integrating with Jason’s control code

Plotting

- Processed on separate computer
- Will integrate to application

Nate Lemons

- Chose parts based on requirements found through research
- Determined ideals for environment variables
  - Worked with Jason to ensure thresholds resulted in desired environmental conditions
- Designed and built irrigation and air flow systems

Matthew Sargeant

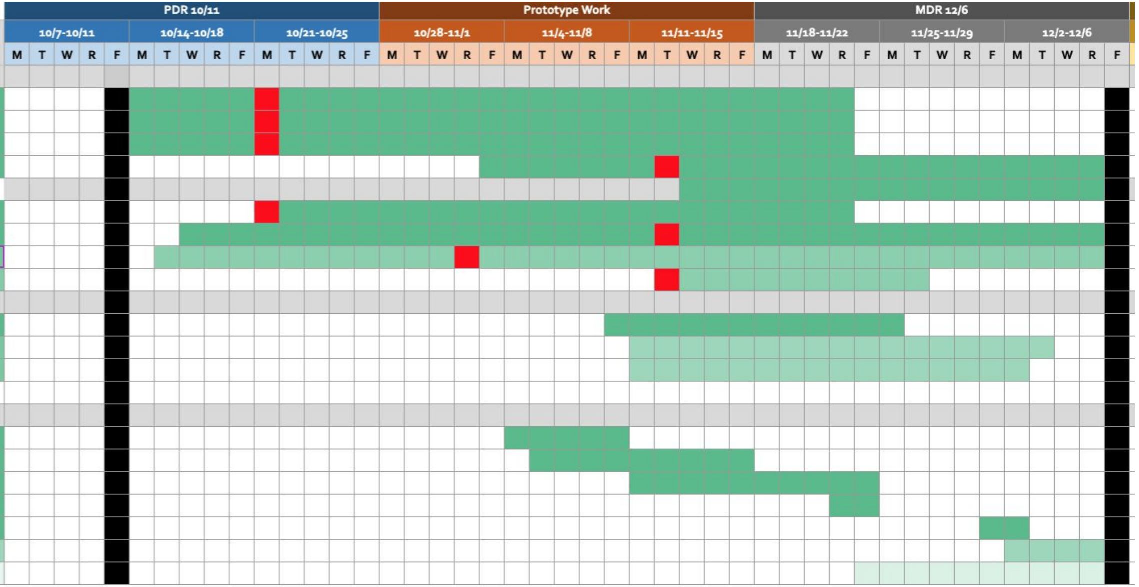
- Built prototype enclosure with Jason
- Determined heating element
- Researched power distribution
- Wired BJT switches
- Choose sensors

Jason Trainor

- Wrote and tested control code
- Created cron job to automatically run programs
- Integrated other’s parts to raspberry pi
- Made calls to plant directory (information on .csv)
- Worked with Matt to construct prototype
- Updated Block Diagram
- Working on Anomaly Detection
- Demo Codes (bench sides and MDR)

Gantt Chart

	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	Sub System 1: Sensors					
1.1	Temperature Sensor	Matt	10/14/19	11/22/19	38	100%
1.1.1	Humidity Sensor	Matt	10/14/19	11/22/19	38	100%
1.2	Light Sensor	Matt	10/14/19	11/22/19	38	100%
1.3	Float Sensor	Nate	11/1/19	11/22/19	21	100%
2	Sub Sytem 2:					
2.1	Heater	Matt	10/21/19	2/24/20	123	100%
2.2	Pump	Nate	10/16/19	11/22	36	100%
2.3	Fans	Nate	10/15/19	11/22/19	37	80%
2.4	Lights (LED)	Matt	11/11/19	11/22/19	11	75%
3	Sub System 3					
3.1	Console Interface	Austin	11/8/2019	11/25/2019	17	100%
3.2	Plotting	Austin	11/9/2019	12/4	25	80%
3.2.1	Logs	Austin	11/9/2019	12/1/2019	22	80%
3.2.2	Data Analysis	Matt	01/01/20	02/24/20	53	0%
5	Control Code					
5.1	Automation of Light	Jason	11/2	11/9	7	100%
5.2	Automation of Temp/Humidity	Jason	11/5	11/15	10	100%
5.3	MDR Control Code (First Logs)	Jason	11/9	11/22	13	100%
5.4	Schedule (Cronjob)	Jason	11/21	11/22	1	100%
5.4	Demo Code	Jason	11/29	12/2	3	100%
5.4	Anomoly Detection	Jason	12/2	12/19	17	60%
6	Power Distribuiton	Matt	11/22	02/24/20	92	15%



CDR Deliverables

- Completely built prototype with 2 levels for 2 different plants
  - insulated material, UV protected greenhouse
- Fully autonomous greenhouse for 2 seperate grow cycles
- Additional sensors and controls for both levels of greenhouse
  - additional LEDs, Fans, pump, sensors
- Bluetooth communication of data to computer or application

Budget

- Greenhouse Unit box ~ $ 75 - 125
- Sensors
  - Hygrometers ~ $15
  - Photosensitive elements ~ $5
- PCB ~$60
- microcontroller 3 x 10 ~ $30
- Materials for growing 15 cycles x ~ $3 =~ $45
- Heating cable ~$9/ft x 2 = $18
- Fans ~ $5
- Lights ~ $30
- Irrigation ~ $50
- Miscellaneous ~ $50
- Total estimate ~ $385-435

Budget Status

- Fan x1 - $11.15
- Adafruit photoresistor x10 - $9.50
- Honeywell Hygrometer x2 - $34.34
- Vinyl Tubing x1 - $5.99
- Sondiko Grow Light x1 - $15.99
- Sub pump x1 - $7.99
- Float sensors x1 - $10.99
- Louver x1 - $7.20
- MicroSD card x1 - $12.59
- Heat cable x2 - $47.76

Total - $163.50
Remaining - $336.50

Demo

UMassAmherst
The Commonwealth's Flagship Campus

Questions?

UMassAmherst

The Commonwealth's Flagship Campus