Developing Intelligent Control and Automated Applications in Bio-hybrid Systems

by

Subhashree Radhakrishnan

Developing Intelligent Control and Automated Applications in Bio-hybrid Systems

Bachelor's Project

Submitted to the Swarm Intelligence Group
in Partial Fulfillment of the Requirements for the
Degree of

Bachelor of Technology

by
Subhashree Radhakrishnan
Peter-Hille-Weg 11
33098 Paderborn

Supervisors:
Jun.-Prof. Dr. Heiko Hamann
and
M. Sc. Mostafa Wahby

Paderborn, July 2016

Declaration

(Translation from German)

I hereby declare that I prepared this report entirely on my own and have not used outside sources without declaration in the text. Any concepts or quotations applicable to these sources are clearly attributed to them. This report has not been submitted in the same or substantially similar version, not even in part, to any other authority for grading and has not been published elsewhere.

Original Declaration Text in German:

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen worden ist. Alle Ausführungen, die wörtlich oder sinngemäß übernommen worden sind, sind als solche gekennzeichnet.

City, Date

Signature

Acknowledgement

First and foremost, I would like to thank the International Office of Amrita and University of Paderborn for giving me an opportunity to be a part of the Student Exchange Program and providing me a platform for learning.

I express my gratitude to Prof.Dr.Heiko Hamann, Professor for Swarm Robotics and Computing Architectures for providing me the opportunity to work in his project group Flora Robotica. I owe my sincere thanks to my mentor M.Sc. Mostafa Wahby, Research Assocciate, Section algorithms and complexity, for his unwavering support, expert guidance and valuable suggestions from time to time throughout my project. I would also like to extend my thanks to my co-mentor M.Sc. Mohammad Divband Soorati, Research Assocciate, Section algorithms and complexity for his valuable guidance.

I would also like to thank the other members of Flora Robotica project Team for their support and help. Finally, I express my deepest gratitude towards Dr.Sasangan Ramanathan, the Dean of Amrita School of Engineering, Dr.Sindhu Thampati, the HOD of Electrical Engineering Deprtment, and all the staff of the Electrical Engineering Department in supporting and encouraging me to be a part of this student exchange program.

Contents

List of Figures

CHAPTER 1

Introduction

The key idea of the Research Project Flora Robotica is to build symbiotic systems of plants and robots in order to establish synergies between them. The goal is to develop and to investigate closely linked symbiotic relationships between robots and natural plants and to explore the potentials of a plant-robot society able to produce architectural artifacts and living spaces [HWS+15]. These bio-hybrids could build mixed societies where humans communicate with plants to form desired structures. The natural plants form structures and provides sensing capabilities while the robots influence the natural growth of plants, their decision making abilities in order to make useful structures out of them. An overview of the concept behind florarobotica is given in figure 1.1.

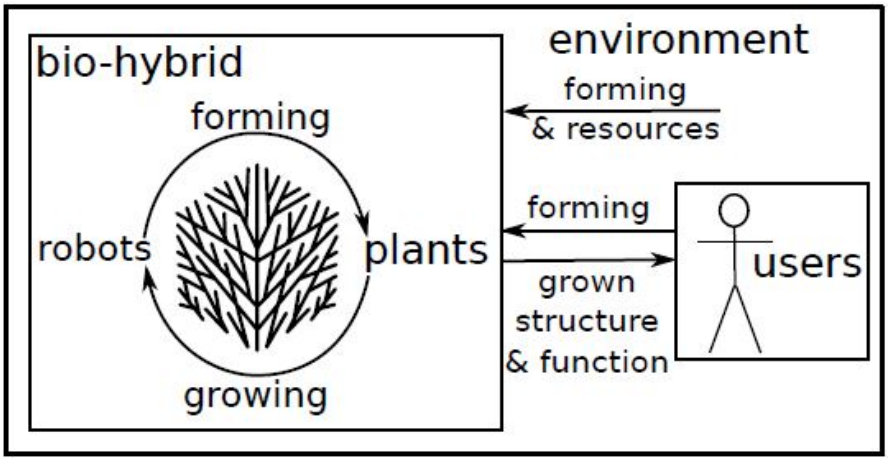


Figure 1.1: The schematic diagram that gives an overview of Flora Robotica[HWS+15]

1.1 Motivation and Ojectives

The primary motivation behind this project is to contribute towards building Bio-Hybrid Systems. The objective was to build automated systems, conduct

priliminary experiment with plants and provide some intelligent control algorithms to extrapolate some data of plant motion and its characteristics.

Automatic Garden Processes such as watering, keeping track soil temperature and providing light for required duration helps in the enhanced growth of plants. Especially, the sleep time for plants is required (light-off period) in order to avoid over drying and hindered growth. Automating such essential processes averts the need of human intrusion to nurture plants and provides reliable control and monitoring. The objective of this experiment is to automate the various processes required to nurture the growth of plants such as light, water and soil temperature remotely. This is implemented with the help of raspberry pi and wireless power outlets[1].

Automation Plant Growth Simulating Natural Plant growth is important to establish a synchronization between artificial structures and the plant growth pattern. Thus this experiment tries to emulate the phenomenon of phototrpism of plants by the use of mechanical artefacts such as springs and motors embedded with light sensors.

Mini-box The prime objective of florarobotica is to integrate plants and non-living artefacts. The factors such as selection of right intensity of light, the type of plant used, the nature of its growth and choosing the right material of the artefacts that needs to be integrated are very important in stimulating plant growth and in enhancing the symbiosis between nature and artificial systems. Hence this experiment aims in deriving useful information about plant, light source, plant growth and to provide autonomous control over the bio-hybrids with the help of Raspberry pi, pi camera and Robotic nodes.

Image Processing Developing Intelligent Controllers to effect artificial stimuli and decision making capabilities in plants help in shaping plants as per human discretion. Many approaches have been deployed and controllers have been implemented previously in florarobotica [ZHS15] [DSH15]. This experiment aims in contributing towards one such controller proposed in [WSMH15]. An algorithm is proposed to find the tip of the plant after experimenting the viability of various algorithms. Also, the motion of tip of the plant was tracked which gave details about the growth pattern.

[1] https://www.amazon.com/Etekcity-Wireless-Electrical-Household-Appliances/dp/B00DQELHBS

CHAPTER 2

Automatic Garden

This chapter gives an overview of the modules used, setup, working principle of Garden Automation. The Garden Automation was carried out using RF based wireless power outlets and humidity sensor.

2.1 Wireless Power Outlets

2.1.1 Remote Controls

The Remote controls that were used in the experiment were Etekcity wireless remote that requires 12V power supply and has a current rating of 10A.

(a) (b)

Figure 2.1: Wireless power outlets (a) etekcity wireless power outlet remotes, and (b) the RF based Etekcity power sockets

The remote initially needs to be programmed by synchronizing the push buttons with power sockets. The programming sequence can be found in the user manual

[1]. This would assign a particular push button on remote to particular power socket and control device. The remote sends RF signals at a frequency of 433.94 MHz in range of about 10 feet. The power sockets also operate at a frequency of 433.94 MHz.

2.1.2 Working Principle of Wireless Automation using ELRO remotes

The ELRO remotes basically work using RF Communication. Each button on the remote when pressed sends a particular 24 bit code that activates the receiver which in turn powers a relay circuit that connects the appliance to 230 V power supply.

The receiving end socket has a decoder that decodes the signal received. For example, the decimal value corresponding to ON 1 of the remote is 21811. It's binary equivalent is 00100001100000010001. The first 16 bits gives the address of the socket that it is synchronized with and the last four bits indicate the 'ON' or 'OFF' state. In this case 0001 indicates ON state.

2.1.3 Implementation of Hardware Based Automation

Setup

The PCB board of the remote is configured and soldered to a transistor. The transistor is used as a switch in order to emulate the press of the button. The overall block diagram of this setup is given in figure 2.2.

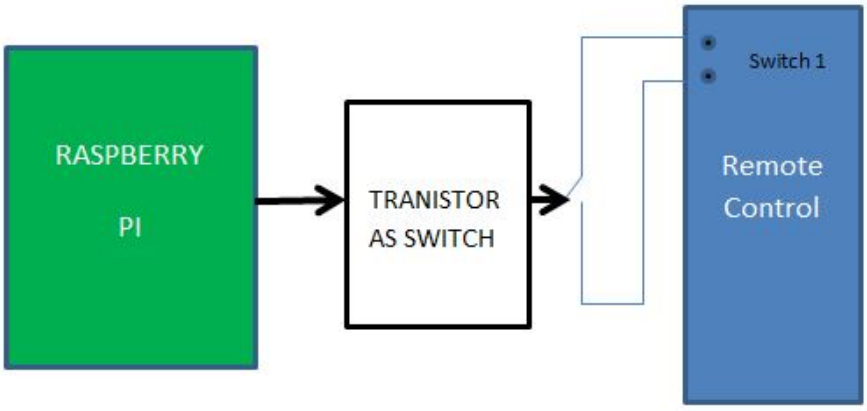
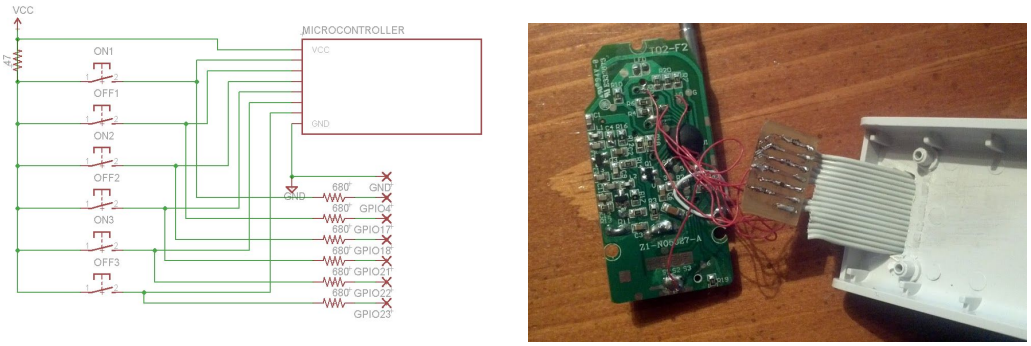


Figure 2.2: Block diagram for the hardware based interface of the wireless power outlets.

[1]http://blog.etekcity.com/wp-content/uploads/2014/01/10-BH9938U-5-manual-self-learning.pdf

The wiring from the two ends of button on the remote PCB are given to the two ends of the transistor which controls the switching action of the button. The schematic from the PCB is shown in figure 2.3a. The remote is used as transmitter in this setup.



(a) (b)

Figure 2.3: The remote buttons interfacing (a) schematic, and (b) the real time soldering of PCB to the microcontroller[2].

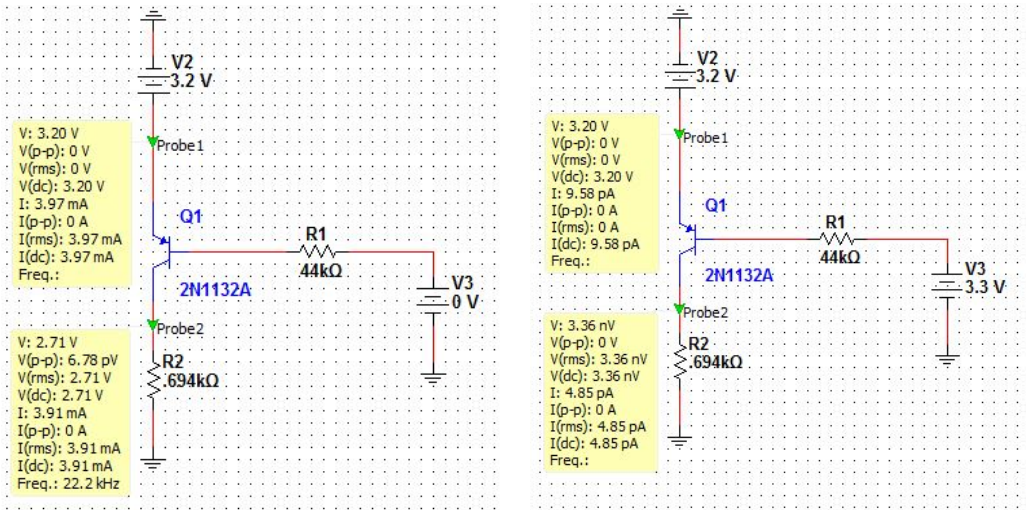
The transistor is implemented in saturation and cut-off regions in order to operate it as a switch. The transistor used in the setup was a PNP transistor. The base of the transistor is given to GPIO pin of raspberry pi. Whenever the base voltage is zero the transistor is in conduction mode and is equivalent to button PRESSED. When the GPIO pin goes high supplying 3.3V, the transistor is in OFF state and is equivalent to button RELEASED. The figures 2.3a and 2.3b show the simulation of this in MULTISIM. As shown in figure 2.4a, when GPIO is given 0 V,the voltage on the second lead of transistor is 2.7 V showing that only 0.5 V has dropped which implies that both ends of transistor are almost at same voltage(conduction).Whereas in the figure 2.4b, the voltage appearing on the second lead of transistor is 3.36 nV which is almost equivalent to 0 V.(non-conducting)

Selection of parameters

A load resistance of 0.694 kΩ was selected by trial method. The resistance was chosen on the criteria of the least resistance between button ends, that could trigger the transmission of signal on button press. The other parameters were consequently calculated as given by equations 2.1, 2.2, 2.3 and 2.4.

$$I_{collector} = \frac{V_{collector}}{R_{collector}} = \frac{3.2}{0.694} = 4.61\,\text{mA} \tag{2.1}$$

$$I_{base} = \frac{I_{collector}}{\beta_{currentgain}} = \frac{4.61}{75} = 0.000\,061\,4\,\text{A} \tag{2.2}$$



(a) (b)

Figure 2.4: The transistor state is (a) ON when supplied 0 V, and (b) OFF when supplied 3.3 V.

$$V_{base} = 3.3 - V_{diode} = 3.3 - 0.7 = 2.6\,\text{V} \tag{2.3}$$

$$R_{base} = \frac{V_{base}}{I_{base}} = \frac{2.6}{0.0000614} = 43\,\text{k}\Omega \tag{2.4}$$

Shortcomings of this setup

1. The connections are messy and complicated as the wires need to be directly soldered to the chip. This process is tedious and has high chances of damaging the PCB.

2. The remote could not be provided a reliable continuous power supply such as using an adapter as the chip was burnt when it was powered with direct 12V from supply instead of a battery. This could be due to the flow of high current which might be greater than the permissible range of the remote.

2.1.4 Implementation of Automation using RF modules

This implementation exploits the RF communication used by the wireless power outlets that was explained earlier in section 2.1.2.

The radio module transceiver pair emulates the function of remote and socket. The transmitter sends radio signal at 433.9 MHz which is inturn received by the rf receiver that operates at the same frequency. The RF module[3] receives serial

[3] `http://www.engineersgarage.com/electronic-components/`

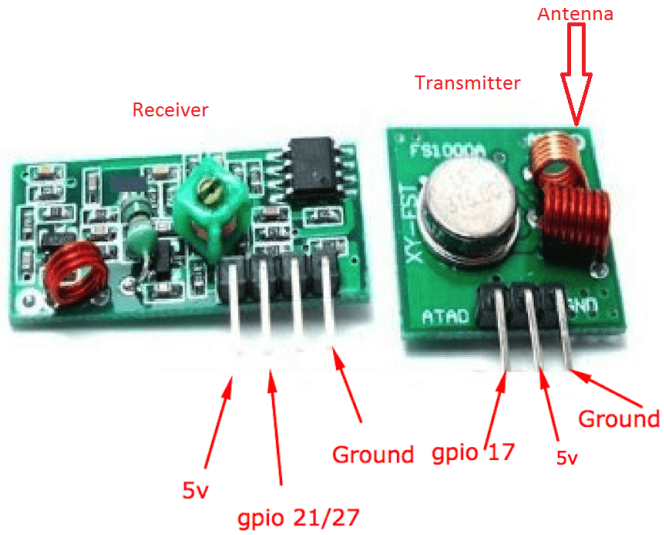


Figure 2.5: The RF module and its wiring specifications.

data and transmits as radio signal at 1 Kbps. The first step is to identify the 24 bit codes associated with each button press. The following steps are followed:

1. The rf module connections are made as shown in figure 2.5.
2. `rfoutlet` library is installed from github repository[4].
3. The codes are compiled and the Rfsniffer program is executed during a button press.
4. The decimal codes corresponding to the buttons appear on the monitor which are noted.
5. These codes are serially transmitted from raspberry pi. This data is in-turn collected by the transmitter and is transmitted as a radio signal. The command used for transmitting a code is shown in listing 2.1

Listing 2.1: The command used for transmitting a code using rfoutlet.

```
$ /var/www/rfoutlet/codesend 21711
```

Advantages of this setup

1. This setup uses less number of wires and thus results in a neat hardware.
2. This doesn't require the remote to be powered by a battery nor an external power source of 12 V. The only powering needed is 5 V for the powering of raspberry pi.

rf-module-transmitter-receiver

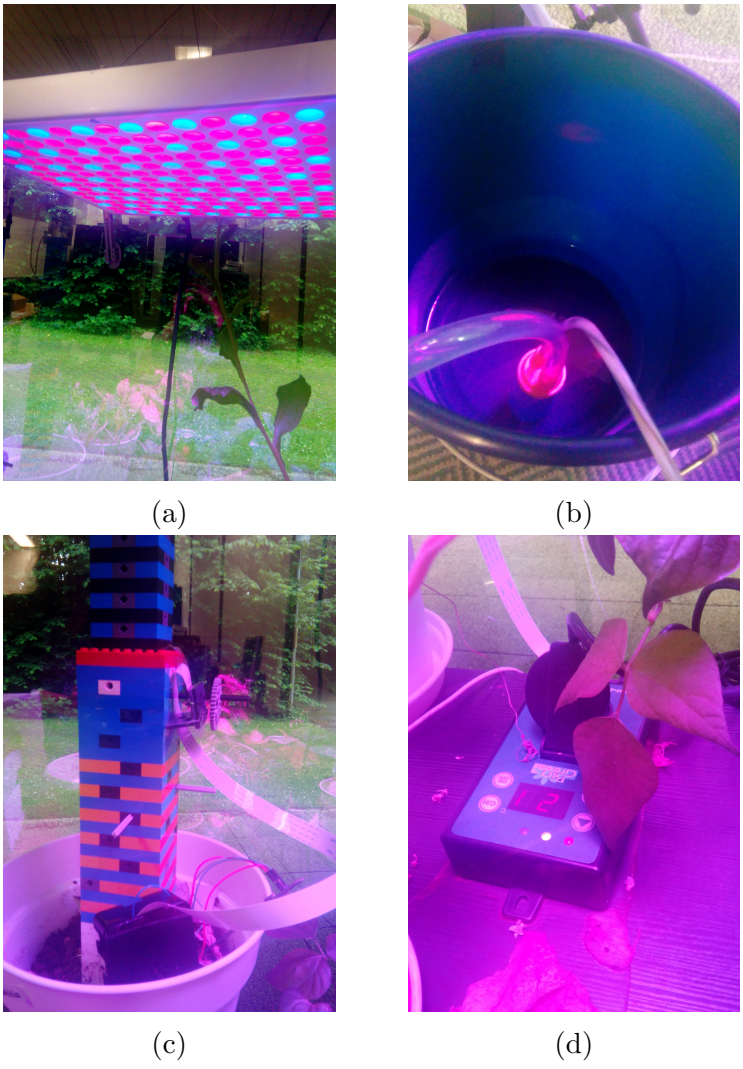
[4]https://github.com/timleland/rfoutlet.git/var/www/rfoutlet

3. Additional power sockets and devices can be added, as the number of devices are not restricted to the number of buttons on the remote. In this setup the remote is no longer used as a transmitter. When the code corresponding to the device is identified, it can be emulated and transmitted using the rf module.

Shortcomings of this setup

The range of transmitter is confined within the room. An additional antenna needs to be attached to the transmitter to increase the range.

The overall automatic garden setup is shown in figure 2.6.



(a) (b)

(c) (d)

Figure 2.6: Automatic garden setup (a) the grow light with day night cycle, (b) the pump set, (c) the camera and (d) the heater automated by raspberry pi.

2.2 Humidity Sensor

The humidity sensor used in the setup was Chirp capacitive soil moisture sensor[5]. This sensor can be interface to raspberry pi either through i2c protocol. This sensor is based on capacitive moisture sensing.

2.2.1 Working Principle and Setup

The moisture sensing by the chirp sensor is based on the value of capacitance that appears between the two copper plates that are coated on either sides of the sensor. Whenever the moisture content of the soil increases, the dielectric strength of the capacitor on the sensor increases. This consequently increases the value of capacitance. Thus the value of moisture level is directly proportional to the value of capacitance.

Figure 2.7: The humidity moisture sensor setup

Moisture content α dielectric strength α capacitance

This capacitance value can be deduced from the moisture sensor by raspberry pi using i2c protocol. SDA and SCL lines of the raspberry pi are connected to the SDA and SCL of the sensor as shown in the figure 2.8.

The SCL line provides the clock signal to synchronize the data transfer across the SDA lines. The other end of SDA lines are pulled high as the I2C output is active low. Since raspberry pi has internal pull up resistance of 1.8kohms ,there is no need of additional pull up resistors. However if more than two sensors are

[5]http://wemakethings.net/chirp/

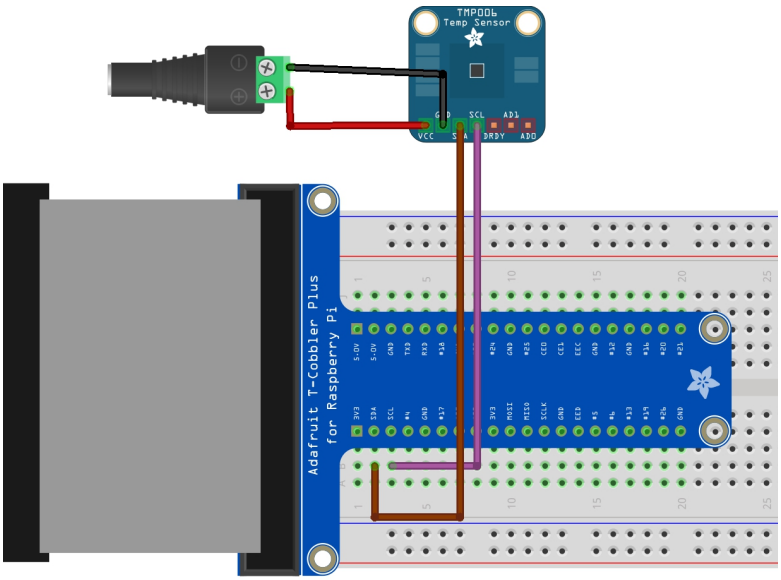


Figure 2.8: The breadboard circuit diagram for connecting an I2C sensor using fritzing.

added to the same raspberry pi(to the same SDA and SCL lines), the total bus capacitance increases and hence additional resistance may be required to maintain the device output high. A pull up resistance of about 3.9 kΩ can be connected from SDA and SCL lines of raspberry pi to the power supply in such cases.

Advantages of the setup

Chirp uses capacitive humidity sensing as opposed to resistive humidity sensing, this means, it does not make an electric contact with the soil, avoiding electrode corrosion and soil.

2.3 Future Practical Implementation

The current setup of humidity sensor is confined to single seedling and a pot. In order to extend this setup to many plants, a common tray could be devised as shown in figure 2.9.

The split seedling tray could be placed over this rectangular tray. Every time when the water is pumped into this tray, the pots absorb water through the holes underneath them, thus ensuring uniform moisture level in all the pots (as the water available to all pots in the tray is uniform). Thus the moisture level can be measured just by using a single chirp sensor placed in any of the pots.

Figure 2.9: The gardening tray.

CHAPTER 3

Plant Automation

This chapter gives an overview of an experiment where the phototropism exhibited by plants was simulated by the use of mechanical artefacts with the aid of raspberry pi. This chapter also explains the prototype of PCB board that was built to interface a light sensor strip.

3.1 The Phenomena of Phototropism

Phototropism is the growth of an organism which responds to a light stimulus. Most plant shoots exhibit positive phototropism[1], and rearrange their chloroplasts in the leaves to maximize photosynthetic energy and promote growth. This results in stimulating the growth of plants in the direction of sunlight.

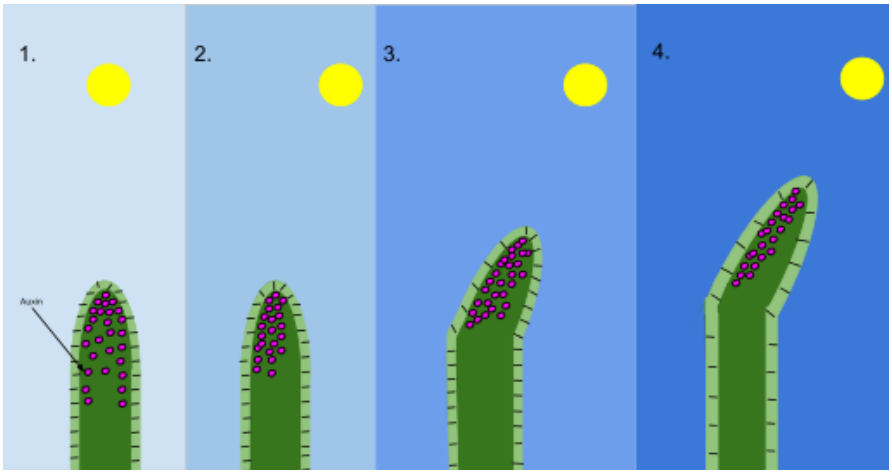


Figure 3.1: Depiction of different stages of phototropism in a plant.

[1]https://en.wikipedia.org/wiki/Phototropism

3.2 Simulation of Phototropism

The main objective of this experiment is to simulate the phenomenon of phototropism i.e. growth of plants towards light by the use of springs, plastic nodes, stepper motors, light sensors and LEDs. An analogy was implemented between the plant structure and the simulation developed as shown in figure 3.2.

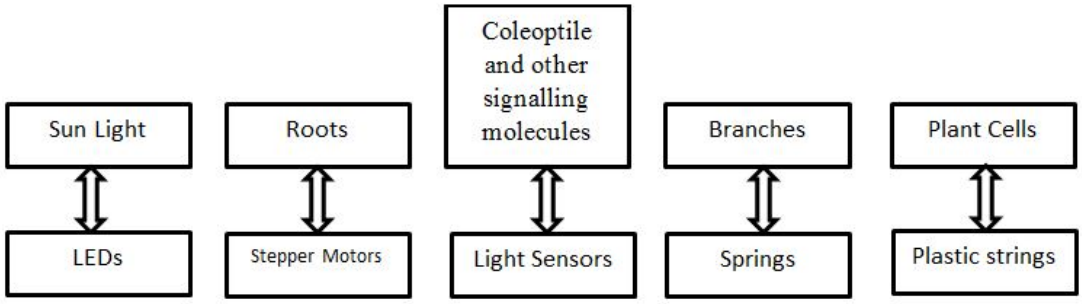


Figure 3.2: Analogy between plants and automation components.

The entire setup consists of seven stepper motors, fourteen light sensors, two LEDs, plastic strings and about six springs. The setup and its orientation is given in figure 3.3. Two LEDs are placed on either sides of the setup given in figure 3.3. This is similar to the source of sunlight for plants. Just as the coleoptile and the signalling molecules senses the light stimuli, the light sensors embedded at the end of every spring senses the light. When the coleoptile senses the presence of light, the cells have a chemical called auxin that moves to shaded regions. This results in elongation of plant cells on other end and leads to growth of plant towards light. Similar to this phenomenon, when the light sensors detect the presence of light they result in the rotation of the stepper motors fitted underneath. These stepper motors are intertwined with plastic strings that are interfaced with the springs. The rotation of stepper motors rotates the plastic strings, which pulls the springs and the structure as a whole rotates left or right depending on the LED that is switched ON.

3.3 Modules Used

3.3.1 GPIO extension

Description

The raspberry pi has a total of twenty six GPIO pins. However, the total number of required GPIO pins exceeds twenty six. The fourteen light sensors require two MCP3008 ADCs as each provides only 8 channels for interfacing analog devices. One ADC requires four GPIO pins and the other ADC will share the SCLK, MOSI,

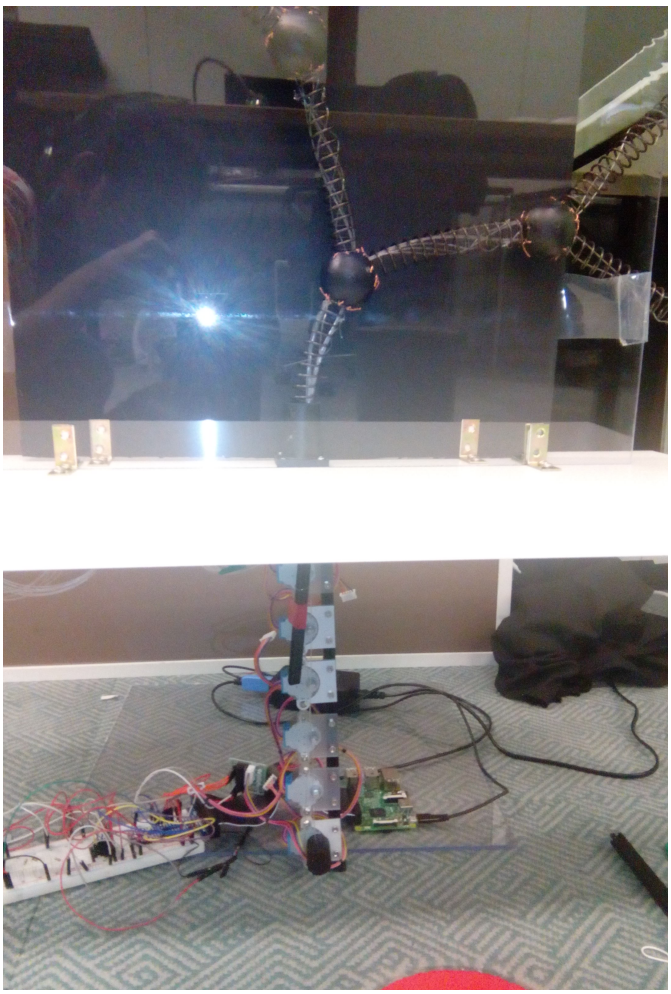


Figure 3.3: The total setup of Automation Of Plant Growth.

MISO and would require one additional GPIO pin for chip select. Hence this sums to five GPIO pins. Each motor requires 4 GPIO pins to be interfaced with the motor driver. Since there are seven stepper motors, this sums to a total of twenty eight. Each LED needs to be switched with the help of a transistor requiring two GPIO pins in all. Thus the total number of GPIO pins required is thirty five. Thus the GPIO pins need to be extended to interface all the components. There are two methods of extending GPIO pins:

1. **Decoder:** A 4:32 decoder can be devised .Based on the four inputs, the decoder gives four outputs. Thus with the use of four pins, eight motors can be controlled. However this does not provide simultaneous control of all the motors. Only one motor can be triggered at a given instance.

2. **GPIO expander:** The GPIO pins of the raspberry pi can be extended through I2C with the help of MCP23017. The I2C bus enables the master slave configuration wherein the raspberry pi acts as the master and the slave

can be a chip or another microcontroller device. The chip used for GPIO expansion is MCP23008 or MCP23017.The former provides an expansion of eight GPIO pins and the latter provides sixteen pins. Since we require thirty five pins in all, we choose MCP23017.

Working principle of GPIO expander

GPIO Expander uses I2C protocol which sends three bytes namely the address of the slave device, the address of registers and the data to be written to the registers. The SCL provides clock signals to synchronize the data transmission through SDA. In order to make a write to the extended GPIO pins, first the address of the I2C slave device needs to be sent. This address is based on the connections of the A2, A1 and A0 pins. The MCP23017 consists of two registers namely IODIR A and IODIR B. Each register corresponds to a set of extended eight GPIO pins.

MCP23017 Wiring

The wiring diagram for MCP23017[2] is given in figure 3.4.

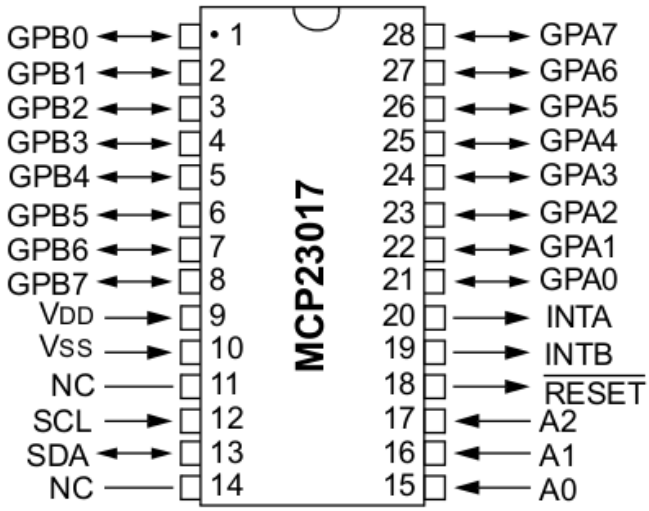


Figure 3.4: Schematic for interfacing MCP23017 with raspberry pi.

The pins marked with GP represent the extended GPIO pins and pins A2-A0 are the address configuration pins. Since the address configuration bits provides a combination of eight addresses, a maximum of 8*16=128(one hundered and twenty eight) GPIO pins can be added to the same I2C bus. The writing to extended

[2] http://www.hertaville.com/interfacing-an-i2c-gpio-expander-mcp23017-to-the-raspberry-pi-using-
html

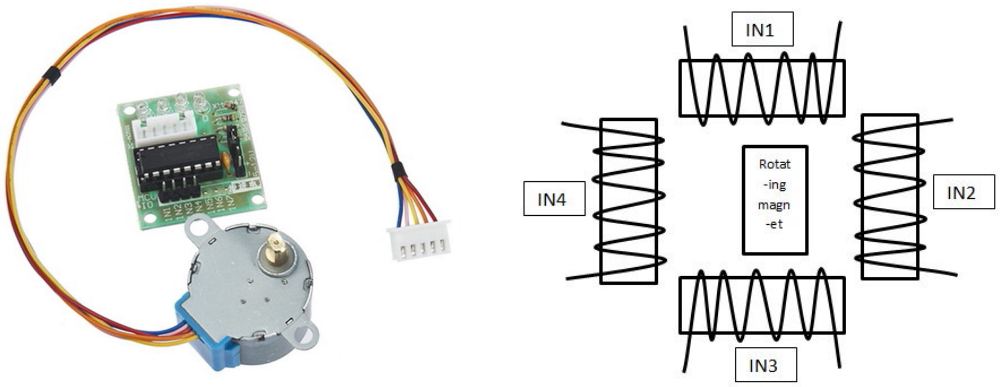
GPIO pins was done using wiring pi 2 library[3]. The following functions were used to operate the motors connected to the extended GPIO pins.

1. `wiringpi2.mcp23017Setup(pin base, i2c_addr)` function initializes the address of the i2c slave to be accessed and the pin base. The pin base is sixty five for IODIR A register and is seventy three for IODIRB.
2. `wiringpi2.pinMode(p, mode)` function configures the pin p as input or output.
3. `wiringpi2.digitalWrite(pin, value)` function assigns the data of HIGH or LOW to the pin.

3.3.2 Motors

Description

The motor used for this setup was 5V Stepper Motor 28BYJ-48 With Drive Test Module Board ULN2003 5 Line 4. Stepper motor was chosen against a servo motor because servo motor provides only 180 degree of rotation whereas we require complete 360 degree rotations for turning the strings and consecutively turning the structure towards a particular direction. The advantage of a stepper motor in terms of power is that the amount of torque can be generated at a safe voltage[4]. The picture of stepper motor used along with driver module is shown in figure 3.5a.



(a) (b)

Figure 3.5: Stepper Motor (a) 28BYJ with driver module, and (b) the internal design.

[3] `http://raspi.tv/2013/using-the-mcp23017-port-expander-with-wiringpi2-to-give-you-16-new-gpio-p`
[4] http://www.tigertek.com/servo-motor-resources/differences-between-servo-stepper-motors.html

Implementation

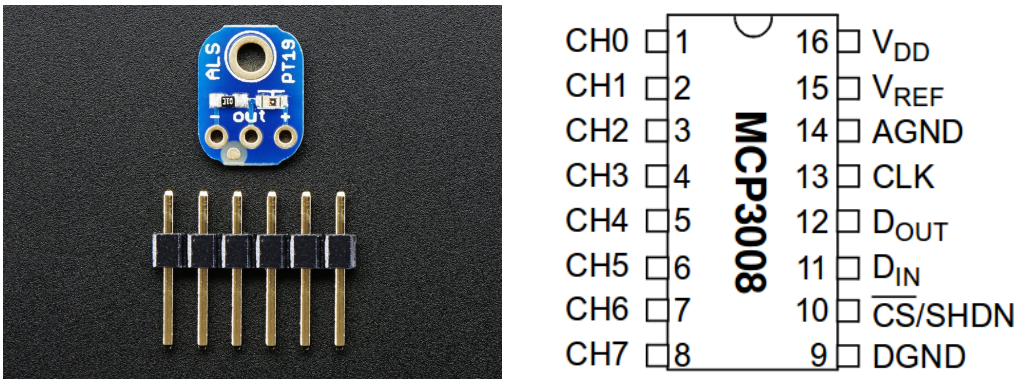
The pins IN1 , IN2, IN3, IN4 of the driver module are connected to four GPIO pins of the raspberry pi.A stepper motor consists of a permanent magnet which is connected to the shaft. This permanent magnet is surrounded by four electromagnets whose one lead is connected to ground and the other is given to IN1 to IN4 respectively. This layout is given in figure 3.5b.

Electric pulses are given to 1N1 —1N4 in sequence to magnetize these electromagnets. As these electromagnets get energized, they attract the rotating permanent magnet, and consequently rotate the shaft. Thus the GPIO pins which are connected to IN1—IN4 are toggled consecutively with ample time delays in order effect rotation of the rotor. The sequence order should be changed in order to change the direction of rotation of rotor and the time delay needs to be changed in order to change the speed of the rotor. The schematic for connection of stepper motor on breadboard is given in fig 3.8.

Light sensors

The light sensor used was Adafruit ALS-PT19 Analog Light Sensor. This sensor is shown in figure 3.6a. Due to the high rejection ratio of infrared radiation, the spectral response of this ambient light sensor is close to that of human eyes[5]. This is one of the advantages of choosing this light sensor.

Since it is an analog sensor, we require an ADC MCP3008 to be connected through SPI to the raspberry pi. Since raspberry pi do not have an internal Analog to Digital Convertor, we use an external ADC chip MCP3008.



(a) (b)

Figure 3.6: (a) Adafruit ALS-PT19 Analog Light Sensor, and (b) the schematic for connecting MCP3008 with RPi.

MCP3008 uses Serial Peripheral interface to communicate with the raspberry pi. The connections of MCP3008 to raspberry pi are done as shown in figure 3.6b.

[5] https://www.adafruit.com/product/2748

The schematic of the connection on breadboard is given in figure 3.8. The SPI requires mainly four connections:

- SCLK pin is responsible for generating clock pulses with which the data transmission between master and slave is synchronized.
- MOSI (Master Out Slave In) pin is responsible for transmitting data from RPi to MCP3008. This transmission is important for receiving the converted digital data from the ADC.
- MISO (Master in Slave Out) pin executes the transmission of converted digital data from ADC to the Master(RPi).
- CS (Chip Select) is important to select the slave to which the communication is to be established.

The digital value can be extracted by bit banging where the clock pulses are manually generated on the pin and the byte is read bit by bit at the positive edge of the clock pulse. The digital value can also be read by using the internal SPI module of RPi along with python modules such as wiring pi and SMBUS.

3.3.3 Foundation Experiments

Motor rotation based on keyboard input

A small test experiment was conducted wherein two motors were made to rotate in the direction as per the input given from the keyboard. The module screen was used to extract the input from keyboard. KEY_RIGHT and KEY_LEFT for controlling motor 1 and KEY_UP and KEY_DOWN for controlling motor two. A wrapper script was used for screen in order to avoid the disorientation of the working terminal after the execution of code. This ensures that a new screen is opened and terminated during the execution of the code. It was programmed such that the motor makes two revolutions in desired direction for a single press of the direction key and the duration of rotation of motor is proportional to the duration of press of the button.

Motor rotation based on light sensor values

In this experiment, the direction of motor is decided by the light sensor values obtained instead of the keyboard input. Two LEDs were placed on two sides along with two light sensors. They were taped together to provide rigidity. The readings from the light sensors were tabulated and every twenty values were summed. The motor was programmed to rotate in that direction of the sensor whose sum of twenty consecutive values is higher. However this logic may not be reliable, as the light status can vary significantly between these twenty values The setup is shown in figure 3.7.

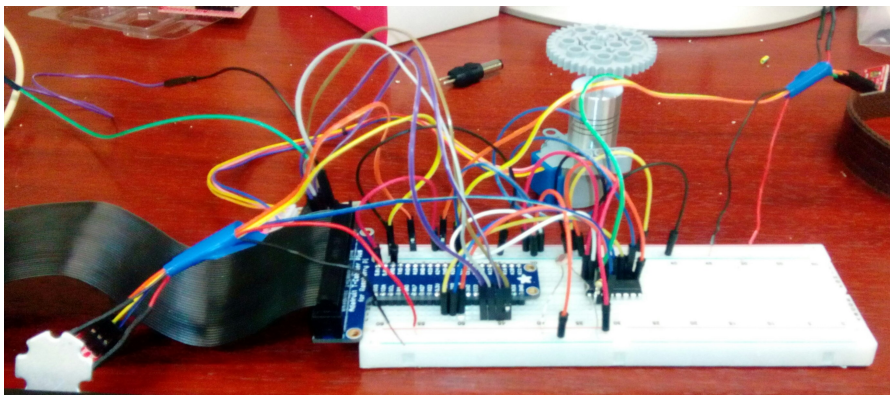


Figure 3.7: The base experiment for motor rotation based on light sensor values

3.3.4 Process Used

Multiprocessing by python was used in order to run the motors parallel. Multiprocessing allows functions to be called simultaneously. Unlike multi-threading, multi-processing does not use shared memory space and does not involve the Global Interpreter Lock(which allows only a single thread to be run at a time). Hence it executes different functions parallel by using multiple cores of the processor. One disadvantage is the high overall memory footprint usage by multiprocessing.

3.3.5 Overall Wiring Diagram

The overall wiring diagram for the setup is given in fig 3.8.

3.3.6 Limitations of this setup

Since motors draw huge amount of current, only a maximum of three motors could be interfaced with a single power source of 5V. Thus a total of three power sources are required to power seven motors making the entire setup robust. The set up on bread board comprises of too many wires and is messy. This can be overcome by soldering the configuration wirings of MCP3008 and MCP23017 to raspberry pi.

3.4 Light Sensor Strip

3.4.1 Description

The future extension of simulation of phototropism on the 2-D space is to form 3-D braided structures. The Thymio robots[6] are going to be programmed to form

[6] `https://www.thymio.org/en:thymio`

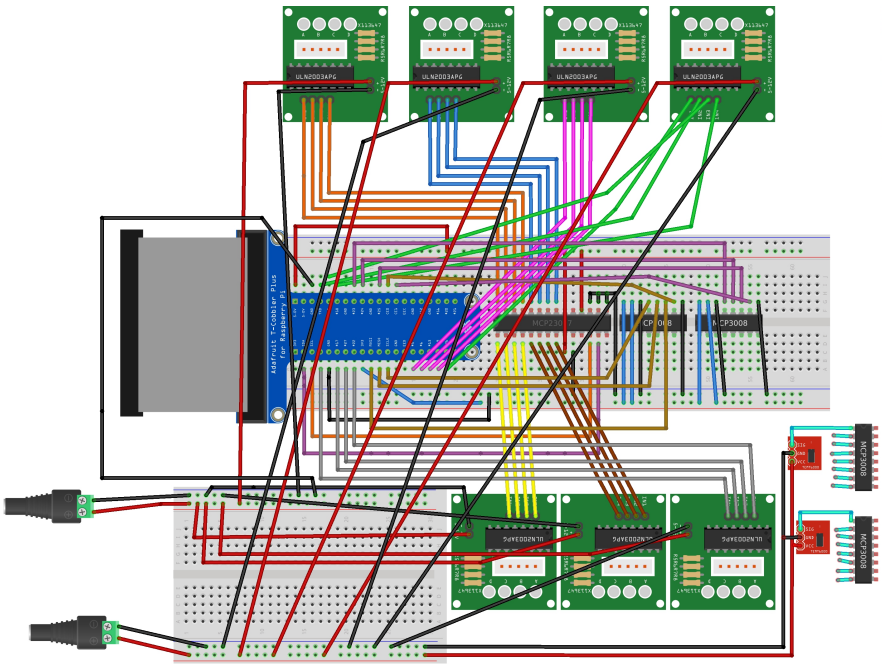


Figure 3.8: The schematic for the entire setup interfacing MCP3008, MCP23017, light sensors and stepper motors using fritzing.

these braided structures. These Thymios will be attached to light sensor strips and will be made to move criss cross in a particular pattern exploiting the line follower capabilities of these bots. In this process,the light sensor strips attached to the thymios will plate with each other and would form a braided structure. These light sensor strips must have maximum number of Light sensors embedded on them. The maximum number of light sensors that can be interfaced with a raspberry pi is sixteen when connected with two ADCs(each ADC provides eight channels).

3.4.2 Prototype

A prototype board was soldered consisting of two ADCs with the Raspberry pi pins. This PCB forms the basis for attaching these light sensor strips to the raspberry pi. Two ADCs can be interfaced to the same raspberry pi with common SCLK, MISO, MOSI but they need to be provided different Chip select pins. The different chip select pin is essential as it identifies the slave with which the raspberry pi needs to communicate. Also at one instance only a dedicated Analog to digital Conversion can be executed, hence there must be a minimum delay of .5 s between two consecutive conversions. Also when the raspberry pi shifts between two ADCs, it needs to deassert one Chip select and assert another Chip select pin. Hence a time delay of about one sec was given between the transition from one ADC to another. Thus sixteen sensors are read over a span of nine seconds

in total. The schematic of the prototype board is given in figure 3.9. The picture of prototype board is given in fig 3.10.

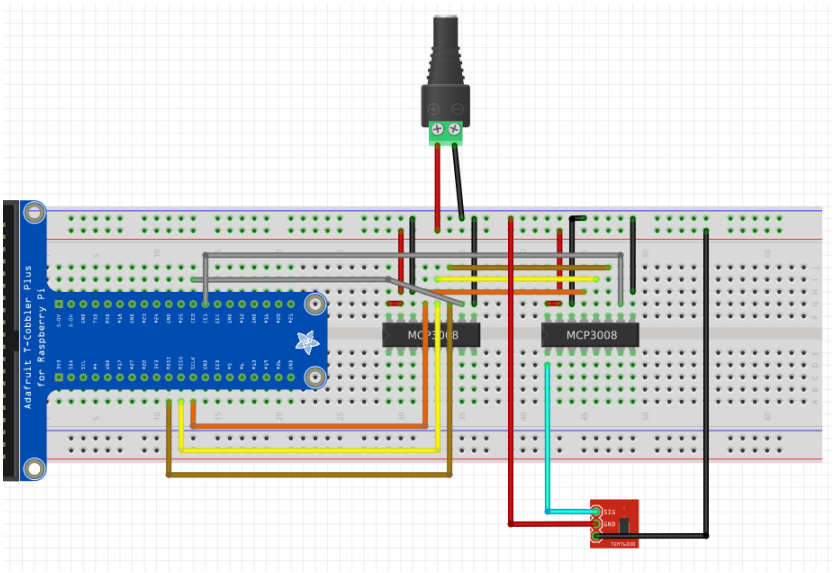


Figure 3.9: The schematic for interfacing 2 ADCs to Raspberry Pi using fritzing.

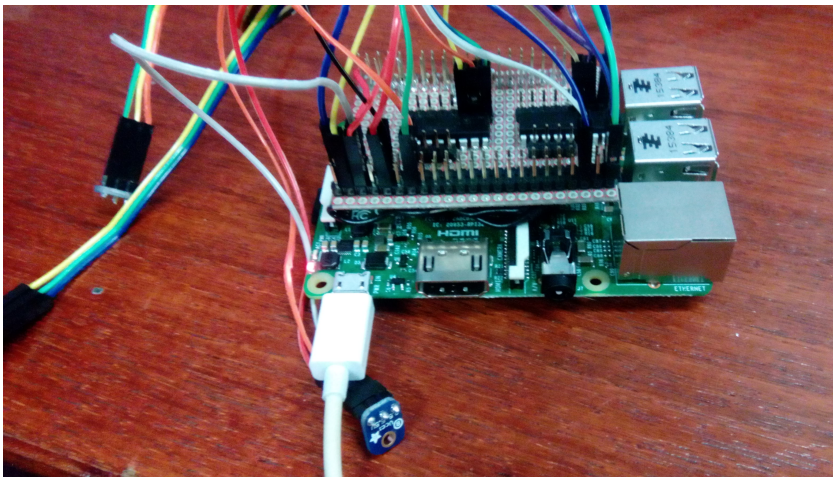


Figure 3.10: The prototype board interfaced with raspberry pi and Light sensors.

CHAPTER 4

Mini Box Setup

This chapter gives an overview of two important experiments that were conducted with climber plants, hanging rods and robotic-node in the black mini box.

4.1 Climber Experiment

4.1.1 Setup

The main objective of this setup was to observe the characteristics, movement and growth of climber beans in the presence of a hanging plastic rod and different types of light sources. Climbers are skinny plants that grab and wind around structures that they come in contact with. The climber plant that was used for the experiment was pole beans. Initially when the setup was experimented with bush bean plant, it was observed that it had grown more number of leaves and failed to wind around the rod. Strangbohen was chosen as it is easy to germinate and grows relatively fast and could wind tightly around the rod. The various components used for the setup are as follows.

Mini-Black Plant Grow Box

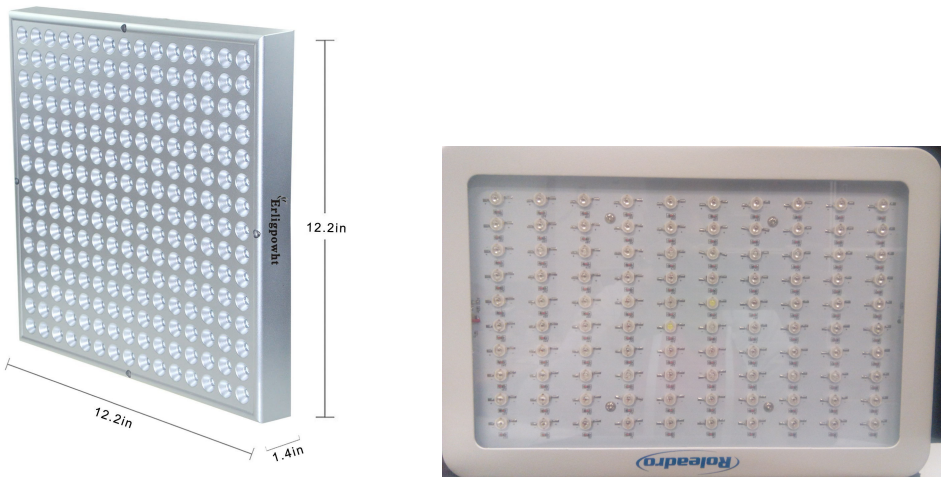
A black box is selected as the environment to nurture the growth of plants. The plant is grown within the box to avert the influence of natural or room lighting and also to prevent the effect of room temperature and air moisture on the growth of plant. Thus the box provides an isolated system in which we have complete control over the lighting conditions, air temperature and moisture content. This helps in determining and observing the effect of various factors on the growth of plants. The figure 4.1 shows the plant grow box.

Figure 4.1: Black Mini Grow Box.

Red and Blue light source

The initial light source used for this setup was 45 W -Red Plant Grow Light Lamp Quad-band Grow Lighting Panel. This colour spectrum was chosen as plants generally absorb red and blue components of white light from sun and reflects green light. Plants that receive plenty of blue light will have strong, healthy stems and leaves. Red light is responsible for making plants flower and produce fruit. It's also essential to a plant's early life for seed germination and root growth[1]. However the pictures taken under this source of light had black stripes and were blurred out with excessive lighting. Hence the light source was changed to 300 W ROHS LED Light fixture. The pictures taken under this source of light had better clarity and less visible black lines. The pictures of both these light sources are shown in figure 4.2.

[1] http://www.gardeningknowhow.com/garden-how-to/info/red-light-vs-blue-light.htm



(a) (b)

Figure 4.2: Grow light (a) 45 W red, and (b) red and blue 300 W RHOS.

Plastic thread rod for Climber Winding

The rod used in the setup for winding of climber plant is 1m threaded rod DIN 975 Plastic. The rod image is given in figure 4.3

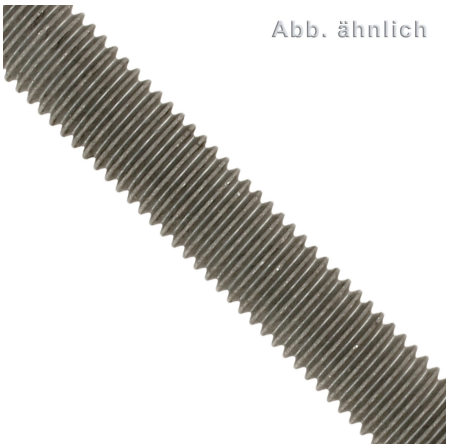


Figure 4.3: Thread Plastic Rod for Climber Plant Support.

Raspberry Pi and Pi Camera

Raspberry pi was chosen as the main processor to operate pi camera and relay module. RPi was used as it has low power consumption, no noise, is compact and has extensive expansion capabilities to accommodate devices through USB or GPIO pins. Raspberry pi camera has 5MP resolution and is connected through

15 pin cable to Camera Serial Interface port on the raspberry pi[2]. The picture taking program takes the experiment name and the delay between two consecutive pictures as arguments. Thus the raspberry pi takes pictures with the delay given and saves them in a folder with the name of the experiment provided.

External Hard disk

A 1 Terabyte WD Elements Hard disk was attached to the fritz box modem which provides a common accessible storage space for pictures.

Relay Module

A relay module was used to control the switching of light source for simulating the day and night cycle. An Opto-isolated 2 Channel 5 V Relay Module was used. The relay module was connected to device as per the circuit given in figure 4.4.

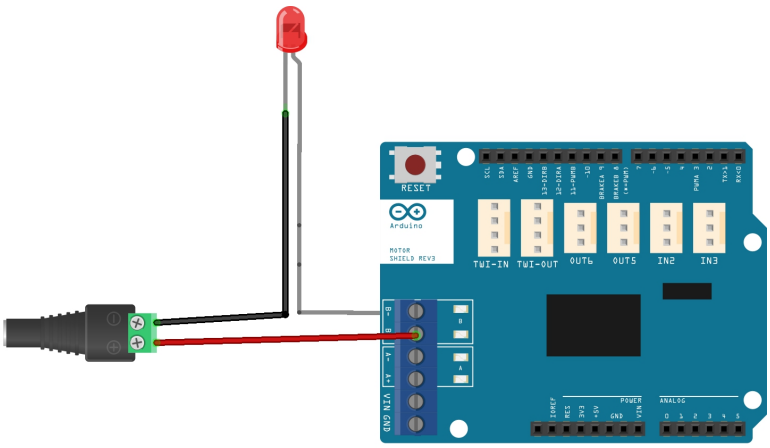


Figure 4.4: Schematic for connection of Relay Module to the Device.

The two leads of the device are connected to the power supply through the relay module. The relay module is inturn controlled by an activator pin which is toggled high or low in the relay to switch the device.

The 'switch' on the relay are of two types namely Normally closed and Normally open. When Normally closed is used, the activator pin needs to be set zero in order to switch on the device. For normally open, the activator pin needs to be set high in order to switch on the device. This module which is an AC solid state relay has the advantage of handling high current flow of AC appliances.

[2] http://uk.farnell.com/raspberry-pi/rpi-camera-board/raspberry-pi-camera-board-5mp/dp/2302279

4.1.2 Processes Used

Flash Drive Transfer

The picture taking program saves pictures in the respective experiment folders in a parent folder named NET. Another folder named store on pi is mounted on a pictures folder in the flash drive. The flash drive transfer program checks the Net folder and creates folders of same name in the store folder and transfers the corresponding pictures from NET to store. This program is executed every 30 minutes using Advanced Python scheduler[3]. Since store is mounted, the experiment folders are shared by store and the pictures folder on the flash drive. Whenever the folder store is not mounted, the program transfers the pictures from NET to another folder named BACKUP. Every time when the program is executed , it first polls the backup folder for pictures and transfers them back to flash drive if the drive has been mounted back. Thus the flash drive transfer program reliably ensures that no picture is missed in case of a network problem and if all the pictures are saved in the flash drive.

Folder Transfer and Time Lapse Video

A folder transfer program was run every day at mid-night from crontab.This folder checks a particular experiment folder (needs to be changed for different experiment in code) and transfers all the pictures to a folder whose name is time stamped with that particular day's date.This program ensures that all pictures are neatly organized and it does not take prolonged time to access a picture in a folder.Another script called videomake creates a time lapse video from the pictures in a given folder using mencoder. The motion tracking of tip of the plant can be observed using these videos.

Crontab and AP scheduler

Cron is a system daemon used to execute desired tasks (in the background) at designated times[4]. The cron jobs can be scheduled by adding an entry to the crontab file as shown in listing 4.1. The crontab files consists of commands specifying the time and the program to be executed at that time.

Listing 4.1: Crontab scheduling entry example.

```
01 04 1 1 1 /usr/bin/directory/python try7.py
```

The first parameter represents the minutes, second represents hour, third represents month, fourth represents day and fifth parameter represents weekday.

Since cron is started by the system and not from shell ,it has very limited environment variables. Hence there are possibilities of certain commands from

[3] http://www.sharpsma.com/webfm_send/1489

[4] https://help.ubuntu.com/community/CronHowto

the script not being executed by cronjob. This needs the environment variables to be explicitly defined by setting the path in the python program or configuring the variables in .bash_profile. Also a boot startup script called Anacron is responsible for running cron jobs as a part of cron.daily. However, the execution of anacron script at boot of system is not reliable[5]. Thus the AP scheduler has the following advantages compared to cron:

1. It is more reliable as it doesn't depend on the system or any other init scripts for it to be executed.

2. A cron job is not executed when the system is shut down. It also doesn't keep track of the jobs missed and executes the job at the next scheduled time instance once when the system is booted. On the other hand, when the python programs are run using APscheduler from upstart, it ensures that the program is executed once every time the system is rebooted after an unexpected crashing.

3. Since the AP scheduler is initialized within the program, it doesn't require the environment variables to be initialized or configured explicitly as in the case of crontab.

Upstart

Upstart is an event-based replacement for the /sbin/init daemon which handles starting of tasks and services during boot, stopping them during shutdown and supervising them while the system is running[6]. Incase of an unexpected crashing or shutdown of system, the upstart ensures that the programs configured in upstart are restarted during reboot. The upstart has an option called respawn which allows the program to be executed in loop infinitely. The programs that needs to be configured in upstart needs to be written as .conf file in init.d . In this experiment, the picture taking program, flash drive program and the sensor logging program were executed from upstart.

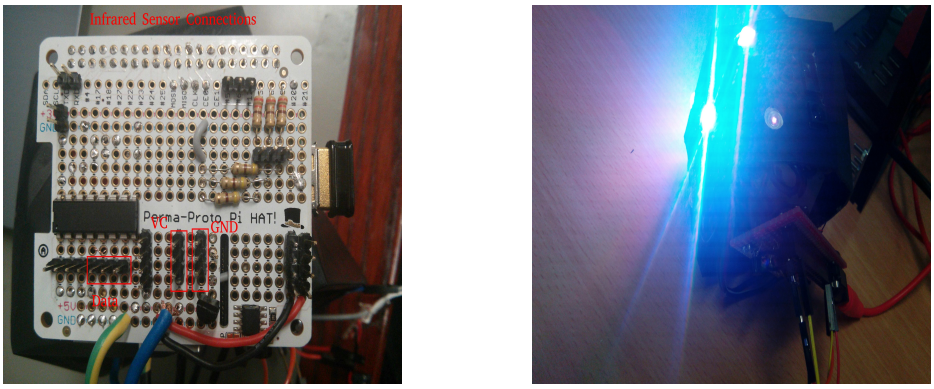
4.1.3 Robotic Node

Description

A Robotic node was designed and fabricated as a part of Swarm Intelligence Research Group to provide wholesome monitoring and measuring of parameters in the bio hybrid systems. This node would form the basis structure to which robotic rods can be attached and would play a vital role in building three dimensional bio-hybrid systems. The node was fitted with IR sensor, photo resistor and pixies. The structure consists of 3 faces each oriented at 45° and each side consists of all

[5] https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=744753
[6] http://upstart.ubuntu.com/

the above mentioned components. The robotic node also provides holes for fitting the rod structures on which the climbers are grown. The robotic node is compact and light weighted and can be easily interfaced with the grow box with the help of Velcro. The robotic node is shown in figure 4.5b.



(a) (b)

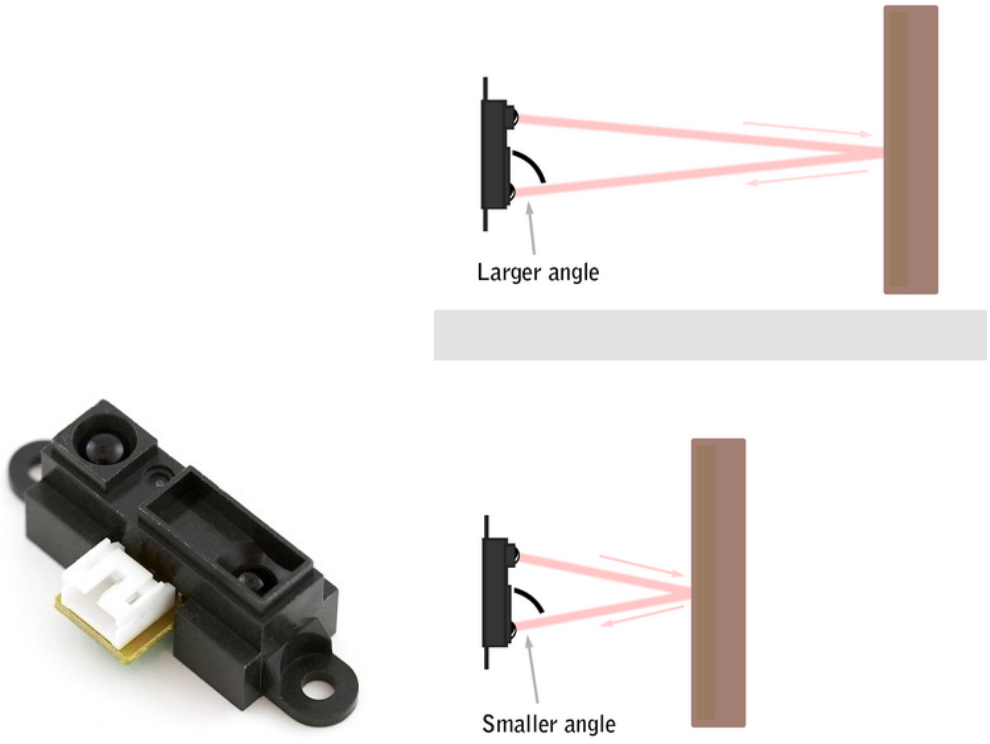
Figure 4.5: Robotic Node (a) the schematic, and (b) the module.

IR sensor

The IR distance sensor works on the concept of transmission and reflection of IR light. The IR sensor used for this setup was GP2Y0A21YK0F 46. It has a range of 10 cm to 80 cm.

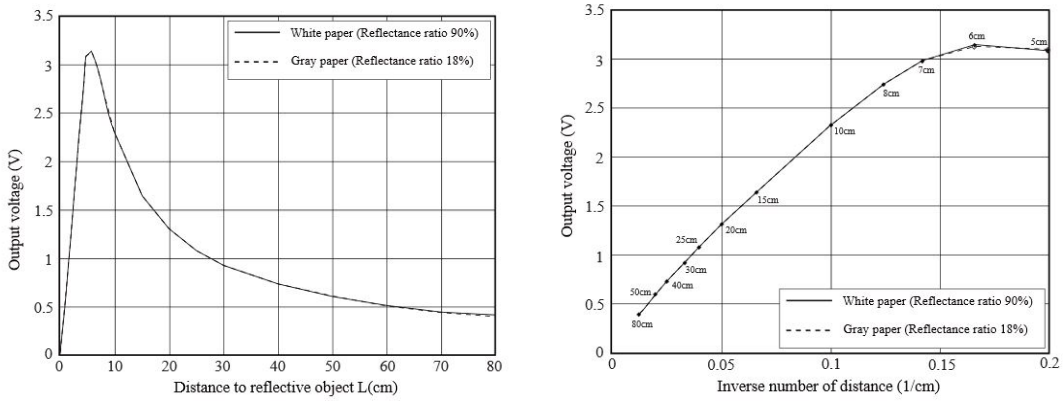
As shown in figure 4.6b, the transmitter and the receiver on the IR sensor are placed at fixed distance from each other. When the IR sensor emits an IR radiation, it is reflected back by an object placed on the way and is received by the Position Sensing Detector. When the object is closer, the angle made by the reflected ray with sensor surface is smaller and this angle is larger when the reflecting object is farther away. The sensor produces a voltage based on the angle. An ADC (MCP3008) whose working was discussed in section 3.3.2, was used to read these voltage values by converting to digital and the distance was deduced consecutively.

Deduction of distance from the digital value: The relation between voltage and distance is deduced from the graphs as shown in figure 4.7.



(a) (b)

Figure 4.6: IR distance sensor (a) GP2Y0A21YK0F 46, and (b) the reflection of IR rays.



(a) (b)

Figure 4.7: Plots of (a) voltage vs. distance, and (b) voltage vs. inverse distance[7].

As it can be seen in figure 4.7a. The voltage and the distance follow an inverse relation. Figure4.8 shows a plot between voltage and inverse of distance which is a straight line. A line of regression was plotted for voltage Vs Inverse distance

and the slope was calculated. This graph is shown in figure 4.8.

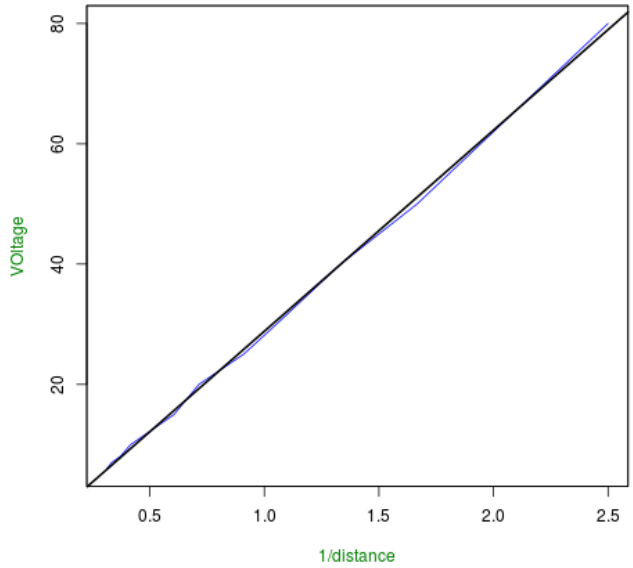


Figure 4.8: Plot of Regression Line for voltage vs. distance.

$$Slope = \frac{2.4 - 1.4}{0.1 - 0.05} = 20 \tag{4.1}$$

$$AnalogVoltage = \frac{DiagitalVoltage}{1023 \times 3.3} \tag{4.2}$$

$$Distance = \frac{1}{AnalogVoltage} \tag{4.3}$$

$$Distance = \frac{6200}{DigitalVoltage} \tag{4.4}$$

The slope is calculated as in equation 4.1. The distance in equation 4.4 is calculated from equations 4.2and 4.3.

Thus voltage vs distance graphs were plotted for climber plant experiment based on equation 4.4.This graph was plotted for the three IR sensors that were placed on each of the three sides which is shown in figure 4.9.

The graph in figure 4.9 was plotted for the corresponding distance of obtained digital IR sensor values over a span of 4 days. The graph was plotted for a data set of about 50,000 values.

As it can be inferred from graph, the distance range detected by the three sensors are as follows:

- IR sensor 1: 10 cm-12 cm
- IR sensor 2: 20 cm-25 cm

- IR sensor 3: 40 cm-60 cm

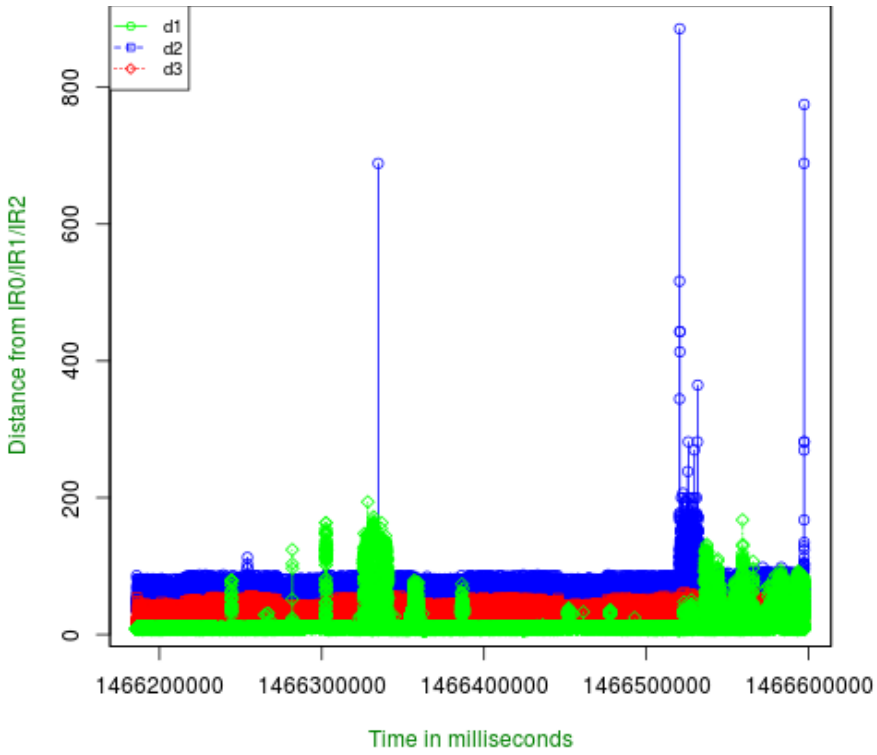


Figure 4.9: The Plot of distances corresponding to IR1/IR2/IR3 against Time in MilliSeconds.

Since IR sensor 1 was placed in that side of the node that was facing the plant, it had detected the presence of plant giving the smallest values of distances. IR sensor 3 was placed facing the left side of the mini box close to it. Hence the IR light was reflected by the surface and it gave an average value of about 25 cm. IR sensor 3 was placed in the front of node facing the door of the mini box which is quite far away from the node. Hence it gave the maximum values ranging from 40 cm. The plot of distances against time for all three IR sensors is shown in figure 4.9.

Photo resistor

A photo resistor as shown in figure 4.10 was also fitted inside the node to detect the presence of light. A photoresistor works on the principle of photoconductivity wherein its resistance changes based on the amount of light it is exposed to. The light excites electrons to conduction band, thus increasing conductivity and decreasing resistivity. The values of photo resistor can be read using an ADC (MCP3008) as explained in section 3.3.2

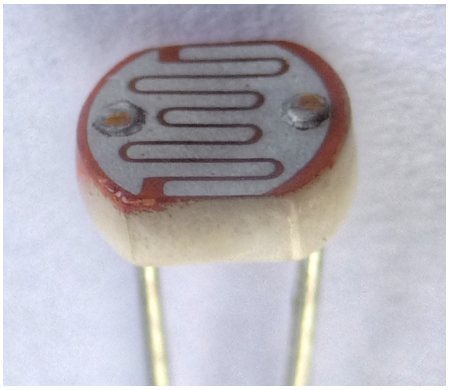


Figure 4.10: A photoresistor.

Performance Monitoring using Net Data

Netdata is a linux daemon, which collects data in realtime (per second) and presents a web site to view and analyze them.[8]. The Net Data provides performance analysis of the raspberry pi processor such as its temperature and the CPU usage. It also provides a web interface which can be accessed with IP and Port which continuously plots these data as graphs. This provides universal access to all computers connected to the same server and provides centralized monitoring of the raspberry pi performance. A screenshot of the NetData performance analysis is given in figure 4.11.

Figure 4.11: Performance Analysis of Sensor Node RPi using NetData.

[8] `https://apscheduler.readthedocs.io/en/latest/`

Live streaming of data using IoT

Internet Of Things is the concept of exchanging data from hardware devices or systems through network with the aid of an IP Address dedicated to the device for communication. Internet Of things was introduced into the bio-hybrid system to establish data over network. Here IoT was utilized to extract the data obtained from the sensors of the node and to consecutively plot graphs of data such as distance from IR sensor, photo resistance value and temperature sensor value against time on an open source web server called Thingspeak.com.

The following steps were followed to configure the IoT live streaming of sensor data:

1. Configure a new channel on Thinspeak.com by entering the fields that needs to be plotted.

2. The API key is defined in the python script that is responsible for reading the sensor values.

3. 3. The Urllib2 python library was utilized to send the continuous data to server. An example code is given in listing 4.2.

Listing 4.2: urlopen example.

```
f = urllib2.urlopen(baseURL +  &field1=%s " % (sensor1 value))
        f.read()
```

The plotting of graphs for IR sensor, photo resistor and temperature sensor on Thingspeak.com is shown in figure 4.12.

Advantages

1. Provides a simple and free solution to continuously monitor the sensor data from anywhere remotely.

2. Plots simultaneous graphs and thus provides an efficient tool for analysis of data and performance.

Disadvantages

1. Since the plotting of values is done continuously, the graphs keep varying and cannot be saved.

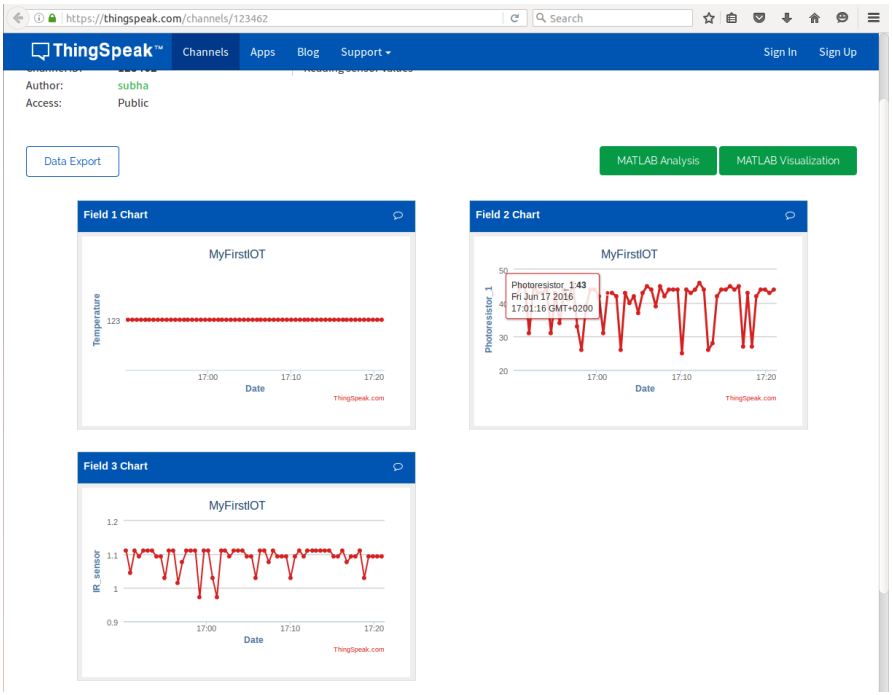


Figure 4.12: Graph Plot of sensor values using IoT.

4.1.4 Results

The Climber Experiment setup with node is shown in figure 4.13.

Figure 4.13: The Climber Winding Experiment Setup.

Thus the observations from the climber experiment are as follows:

1. The pole climber plant had more winding capabilities than bush bean plant.

It also grows comparatively less number of leaves.

2. The pictures taken under Red and Blue 300 W source of light had more clarity and less visible black stripes.

3. The germinated pole climber took only a week's time to wind around the hanging rod.

4.2 Rod Movement Experiment

4.2.1 Description

Phototropism is the tendency of the plant to respond to light stimuli and grow towards the light source. Thigmotropism is the tendency for a plant organ to bend in response to touch[9]. This experiment was setup to exploit these two phenomena of pole climber to ultimately move a hanging rod. The layout of this setup is given in figure 4.14.

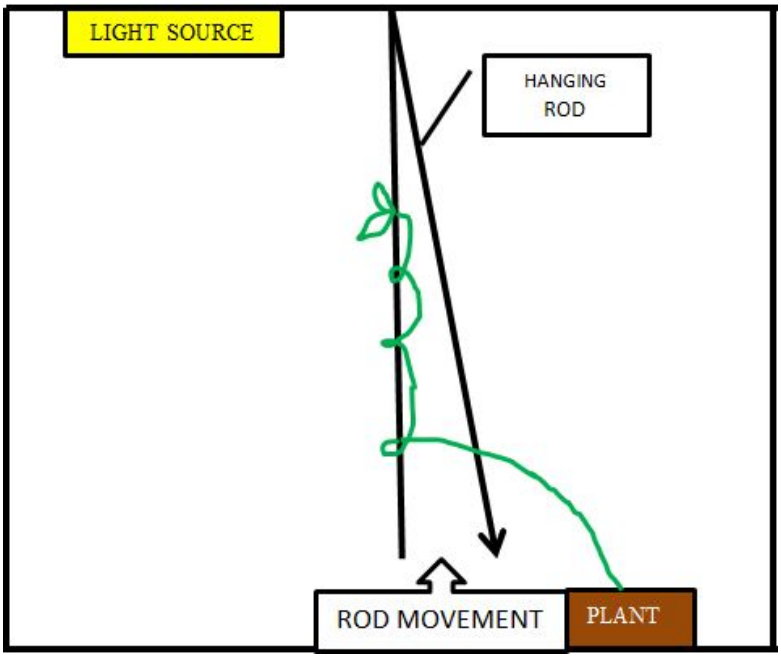


Figure 4.14: The Rod Movement Experiment Layout.

The plant pot is displaced a little away from rod instead of being placed directly under the rod. The plant begins to grow inclined towards the light source owing to phototropism. In the process, when it touches the hanging rod, it grabs the rod and would start winding around it owing to thigmotropism. The proposed system expects the plant to grab the rod and pull it in the process. A raspberry pi camera

[9] http://www.biologyreference.com/Ta-Va/Tropisms-and-Nastic-Movements.html

was set near the end of rod to observe its movement. The main objective of this proposed system was to exercise control over the non-living artefacts by plants. Thus the experiment aims in building bio-hybrid system to produce movements in non-living objects.

4.2.2 Observation

The actual result of this setup is as shown in figure 4.15. As seen in the figure, it was observed that the climbers could grow to a very large height without the support of the rod. Also since the box was comparatively small, the light source was almost uniformly available throughout the box averting the need for the plant to bend excessively to access the light. Thigmotropism occurs only when the plant comes in contact with an object. Since the plant has not bent much ,it did not come in contact with the rod for a long while. As seen in the picture ,by the time it has come in contact with the rod, the plant has grown tall enough and failed to exert a pull force on the rod.

4.2.3 Conclusion

Thus this experiment proved that source of light was of more importance than a supporting structure for a climber since the plant was able to thrive for a long length without a support. The next iteration of this experiment can be carried out in a wider box where the source of light is placed much away from the plant. By this way it can be ensured that the plant bends sufficiently and grabs the rod at a much earlier stage. Also the node used provided light from pixies that operated at 30 % power. Since the node was not provided with cooling fans, the pixies were flickering intermittently due to overheating. This could have hindered the growth of climber plant. Hence the next experiment would be fitted with nodes that are interfaced with cooling fans thus ensuring continuous and reliable source of light. Since each climber plant is flimsy, a group of climbers may result in exerting a force on the rod and effecting a movement.

Figure 4.15: The actual Rod Movement Experiment with climber bean and Robotic Node.

CHAPTER 5

Image Processing

The previous chapter dealt with the various bio-hybrid experiments of the pole climber plant. This chapter deals with the processing of images obtained in the minibox bio-hybrid experiments conducted, to make interpretations of plant growth.

5.1 Tip Detection

Observing and monitoring the tracking of tip of the plant plays a crucial role in understanding the growth pattern of climbers. Hence the main objective of this section is to deduce the tip of the plant using image processing techniques.

As discussed in [WSMH15], one of the primary motives of florarobotica is to promote natural growth processes with artificial growth processes. This paper presents a method of controlling plant growth with the help of intelligent robotic controllers. The tip of the plant is given as input to the ANN which determines the corresponding light that is switched ON. This value of $(x,L)_t$ is now passed as input to the controller which determines the next tip position of the plant. The objective of the work was to maximize the growth of plant and make them grow in desired pattern. Here the tip of the plant was deduced by identifying the point with maximum concentration of bright points in the grey scale image of the plant. In this chapter, a new algorithm has been proposed to deduce the tip of the plant in an attempt to provide a faster and more accurate way of deducing the tip of the plant using image processing.

5.2 Proposed Algorithm

In this work, a new algorithm was proposed for deducing the tip of the plant. The flow chart for this algorithm is given in figure 5.1.

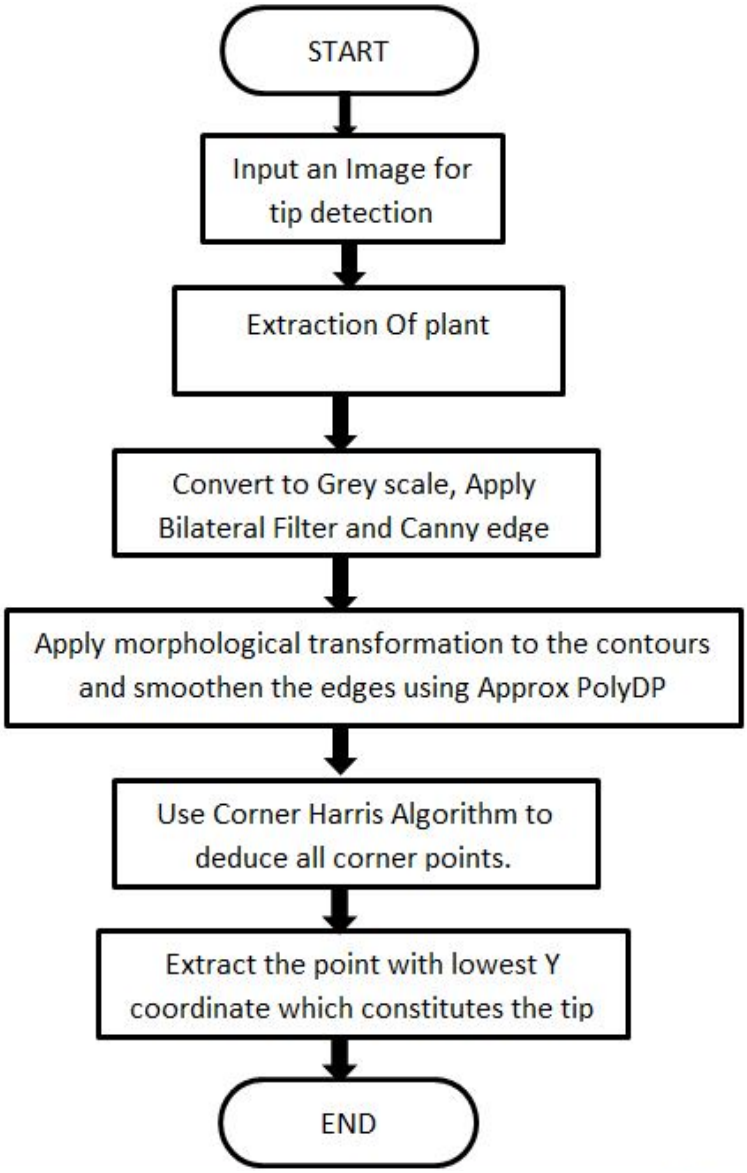


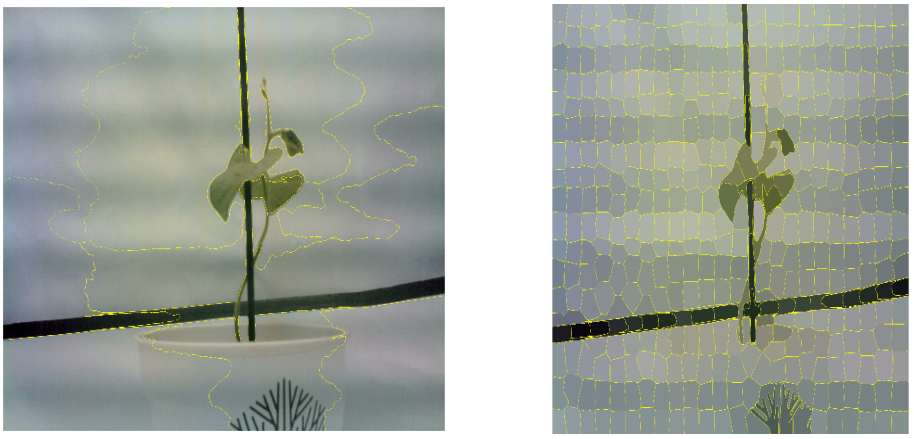
Figure 5.1: Flow Chart of Proposed Algorithm

5.2.1 PLant Extraction

Superpixel Algorithm

Super pixel segmentation provides a method to deal with and process a group of pixels rather than individual pixel. This grouping is done based on similarity in colour or texture. The objective was to extract the plant as one super pixel segment. One of the major advantages of Super Pixel Algorithm is its faster computation. The following steps were experimented in obtaining the plant as a single super pixel segment.

a. Segmentation The initial segmentation was performed with Simple Linear Iterative Clustering(SLIC)[ASS+12]. This performs grouping of pixels in CIELAB(5D) color space. Since SLIC performs segmentation in 5D space, the processing is efficient and accurate. Also SLIC provides segments that have smoother and polygonal curves compared to other segmentation algorithms such as quick shift and graph-based segmentation. Scikit SLIC command takes number of segments and the sigma as arguments. Sigma defines the width of the Gaussian Filter kernel[1] which smoothens the image by reducing noise. Greater the width ,implies greater extent of blurring the image .Since only the plant region is required and other intricate aspects of the image are of less importance, the blurring can be high but not too high as it can distort the image. Hence the value of sigma chosen was five.



(a) (b)

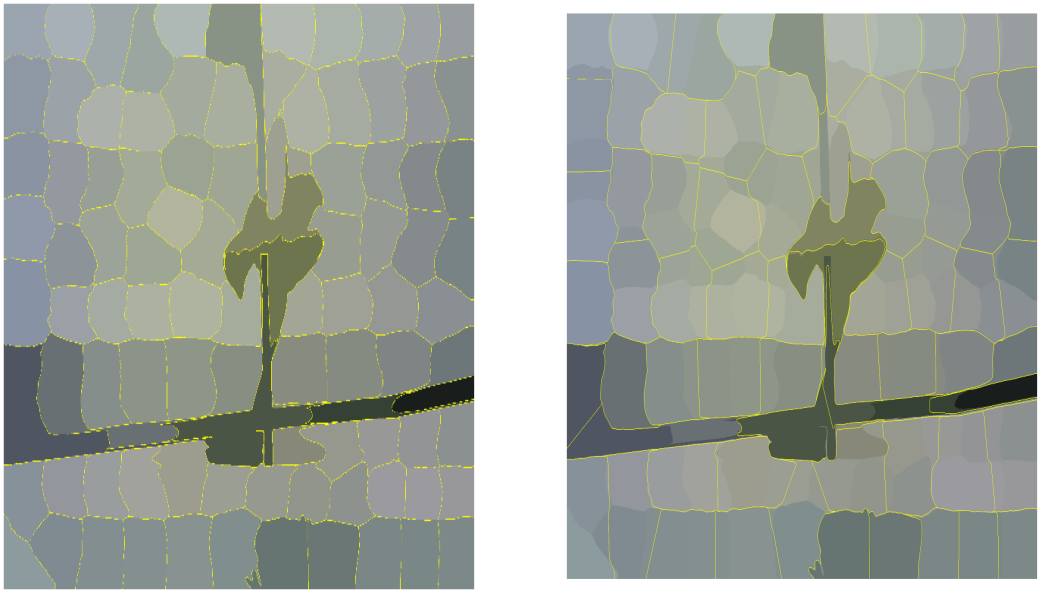
Figure 5.2: Segmentation (a) 10 segments, and (b) 450 segments.

5.2a gives the segmentation with 10 segments and 5.2b gives segmentation with four hundred and fifty segments. As observed from the figures, greater number of segments provide well defined edges. However, over segmentation introduces unnecessary boundaries and redundant segments. Hence based on trial and error, hundred was chosen as the number of segments.

b. Thresholding and averaging Sci-kit image module was used for coding and processing these super pixel segments. Each pixel belonging to a particular super pixel segment is given a unique number and is identified as a label. Various label functions are provided by scikit image for processing these segments. The function used was label2rgb(). The main objective of this was function was to merge

[1] `https://en.wikipedia.org/wiki/Gaussian_blur`

similar coloured regions. The parameter that was used in the implementation for label2rgb was avg. This parameter accesses each super pixel segment, computes the [R G B] values of average colour of that super pixel segment and would converts all pixels to this RGB value. This averaging was done twice and segmentation was applied again. The final result is shown in figure 5.3b.



(a) (b)

Figure 5.3: Averaging (a) single time, (b) twice

A Region Adjacency Graph was constructed over these segments using graph.rag_mean_color. This graph helps in computing the edge weights. The other parameters of this function were mode and sigma. The mode selected was similarity and this mode computes the edge weight based on how similar the average colors of two regions are. The equations governing the weight between two adjacent regions are as follows:

$$d = |c_1 - c_2]| \tag{5.1}$$

$$e^{\frac{-d^2}{sigma}} \tag{5.2}$$

Hence greater the value of sigma, lesser will be the weight between two regions and greater is the possibility of considering two regions similar. The sigma value chosen for the setup was ten. Once when the edge weight has been calibrated, a

function called graph.cut_threshold was used to combine the regions whose edge weight was less than the given threshold. In place argument was set so that the edges that weigh below a particular threshold are combined. A threshold of fifteen was used and the result is shown in figure 5.4

Figure 5.4: Thresholded image at a value of 15

c. Extracting the green coloured super pixel segments The thresholded image consists only of three super pixel segments which needs to be made one. Hence those super pixel segments that were green were extracted by overlaying through all segments. The function color2rgb was used with type parameter as overlay. The overlay fuction was first tested on the original image and result is shown in figure 5.5 This was the result as white was used as the background and there was reflection of plant on the background and many regions were considered under the category green predefined in color dictionary of sci-kit module. Hence super pixel segmentation did not give favourable results due to which this algorithm was

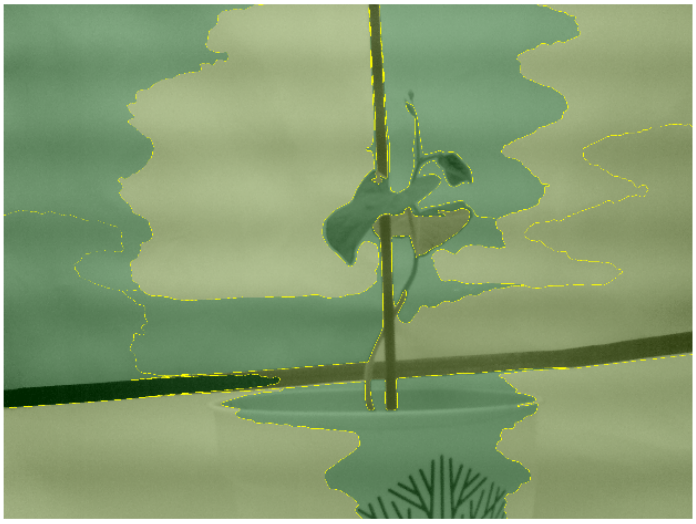


Figure 5.5: overlay image for green segments

not used in the final proposed model. The following enumerates the drawbacks of using this method for plant extraction.

Drawbacks

1. One of the drawbacks of Super pixel segmentation is that it tends to over segment an image and hence the edges may be lost in the process.

2. In the process of merging, it averaged most of the regions to green and significant shape of the plant was lost.

3. The inbuilt range for green identified most of the regions as green.

Owing to above reasons, superpixel algorithm is not the best way to extract plant from image.

Grab Cut Algorithm

This algorithm was tried in order to extract plant from its background. Grab cut algorithm basically helps in extracting the foreground images from the background. The background and foreground matrices are defined initially and the algorithm uses this as a database to train the GMM MODEL. Depending on the data we give, GMM learns and create new pixel distribution[RKB04]. It classifies and lables pixels as foreground or and background. The result for grab-cut algorithm is shown in figure 5.6.

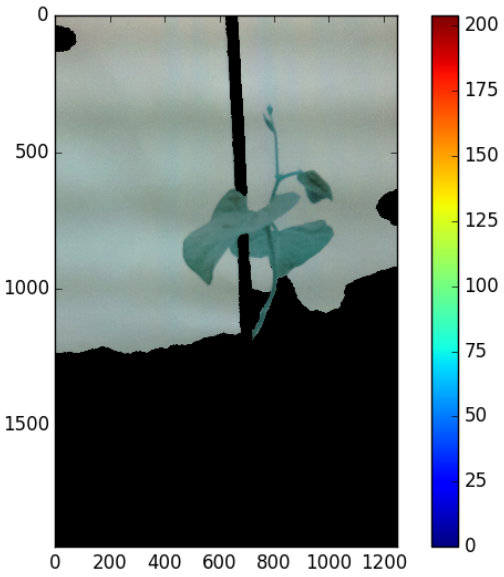


Figure 5.6: The resulting image after grab cut algorithm

Drawbacks As seen in figure 5.6, sometimes the some of the background is also misinterpreted as foreground. The image needs to be manually replaced with corresponding pixels to obtain the accurate foreground image. Thus this algorithm is not suitable for isolted extraction of plant.

Colour Detection

Open cv provides a function called inrange which allows boundaries to be fixed for extracting a set of pixels if they lie within that range and forming a mask using those pixels. This function can be used to define the R, G, B range for green color that defines the plant in the experimental setup. The R, G, B range used for the setup was from [17,70,100] to [70,255,255]. The result of extracting the plant from plant is shown in figures 5.7a. The mask of image that laid between defined boundaries was converted to gray scale and is shown in figure 5.7b.

5.2.2 Applying Filter and canny edge detection

Various filters were applied to the extracted plant to remove noise. This is done by convoloving image with low pass filter kernel so as to remove high frequencies in the image[2]

[2] `http://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html`

(a) (b)

Figure 5.7: Extraction of Plant (a) Replacement of green with white pixels, (b) Grey scale image of extracted plant mask

Blur

This convolves the image with a box filter and averages all the pixels under a kernel area.

Figure 5.8: The resulting image after applying blur

As it can be seen in figure 5.8, the tip of the plant is not visible and hence this filter is not suitable.

Median Blur

Median value of all pixels are taken and all pixels are replaced with this value. The result of this filter is shown in figure 5.9.

Figure 5.9: The resulting image after applying median blur

As observed in figure 5.9, the tip of the plant is almost not well defined and hence use of median filter was discarded.

Bilateral filter

This applies one gausian filter on space and another on intensity difference. While the former ensures that adjacent pixels are considered for blurring, the latter ensures that only pixels belonging to the main element is considered for blurring. Since edges have very high intensity difference from the main element the second gausian filter ensures that edges are not blurred. This way bilateral filter preserves the edges making it the suitable one for the proposed algorithm. The parameters used were five for sigma and twenty for defining distance space. As it can be seen in figure5.10, the tip of the plant is retained unlike other filters.

Figure 5.10: The resulting image after applying Bilateral Filter

Canny Edge Detection

After removing noise, canny edge detection[3] is applied to enhance the definition of edges. In canny edge detection, initially edge gradient and direction gradient is found for every pixel. The direction of edge gradient is perpendicular to edge. Hence the local maxima of pixels in the direction of gradient is considered as edge pixel. This representation is shown in figure 5.11. After deciding on edge pixels,

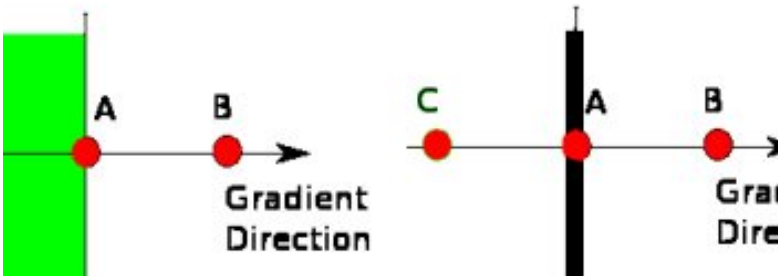


Figure 5.11: Image representing the edge gradient and pixel on edge

the edges are thresholded based on max and min value. The edges with intensity gradient greater than max value are classified edges. Those whose intensity gradient lies between max value and minimum value are classified as edge based on their connectivity. The parameters of max value and min value for function cv2.canny() chosen were one and thirty five.

[3] http://docs.opencv.org/master/da/d22/tutorial_py_canny.html

Figure 5.12: Resulting image after canny edge detection

5.2.3 Adaptive thresholding and Morphological transformation

Adaptive thresholding is done based on a threshold that keeps varying for different regions. The concept of thresholding is that if the value of pixel is greater than a particular value, it is assigned to pixel value of white or black. Here the thresh was calculated by summing the gausian windows of neighbouring regions. The result of adaptive thresholding can be seen in figure 5.13. This has made the boundary

Figure 5.13: Image after adaptive thresholding

more well-defined. Consecutively, morphological transformation was applied to this image. This transformation basically involves erosion followed by dialation [4]. Erosion removes white noise and reduces the size of the image. Dilalation increases the white region of the image.The function used was cv2.morphologyEx(). The result of this is shown in figure5.14.

Figure 5.14: Image after morphological transfromation

5.2.4 Corner Harris Algorithm to deduce tip

After the image was morphologically transformed, the contours of the image were found and Approxpoly DP was applied to the contours to make the boundaries of the contour as fine lines. The corner detection algorithm that was applied was C. Tomasi Corner detection[TK91] which is just a revised version of Corner Harris corner detection. The basic logic behind corner harris is to find those regions that have maximum intensity variation in all directions. This is done by calculating the intensity difference for a particular displacement (x,y). The function used for this algorithm was cv2.goodFeaturesToTrack. The parameters that were given to this function were number of corner points, quality of the points plotted and the minimum euclidean distance. The parameter values used were 100, 0.001 and 7 . When the value for Euclidean distance was two, the tip was identified at a lower point. This was observed even when the total number of corner points was made too less. This shows that the Euclidean distance and number of corer points plays an important role in deducing the tip properly. A higher value of both these parameters provides precise location of tip.

[4] http://docs.opencv.org/3.0beta/doc/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html

(a) (b)

Figure 5.15: The tip detected (a) wrong tip for euclidean distance=2, (b) correct tip detection of plant

5.3 Motion Detection

Motion Detection of the Climber plant was the next objective. Observation of motion of tip of the plant is important for building next iterations of bio-hybrid systems. It was observed from previous experiments that tip of climber plant follows a very random path by swirling and doesnt grow in a straightforward manner. Hence it is important to observe and track its motion. Though IR sensors fitted in the node provides a comprehensive picture of the distance of plant tip at different instances, it does not provide any information on the details of the path followed by the plant tip.

5.3.1 Differential method

Motion Detection was initially tried by absolute differencing of continuous images. The difference was checked with a threshold and was classified as movement if it was greater than a particular threshold. However this method failed to work as the motion of plant tip was very intricate and its nuances could not be detected.

5.3.2 Motion tracking using proposed algorithm

The proposed algorithm for tip detection was applied to a series of images taken in mini-box setup over a span of one week. The tip was deduced for each image

and circles representing the tip of plant in each of the image was plotted in the same final image with the tip of the final image marked in green. The image representing this movement of plant tip is shown in figure 5.16.

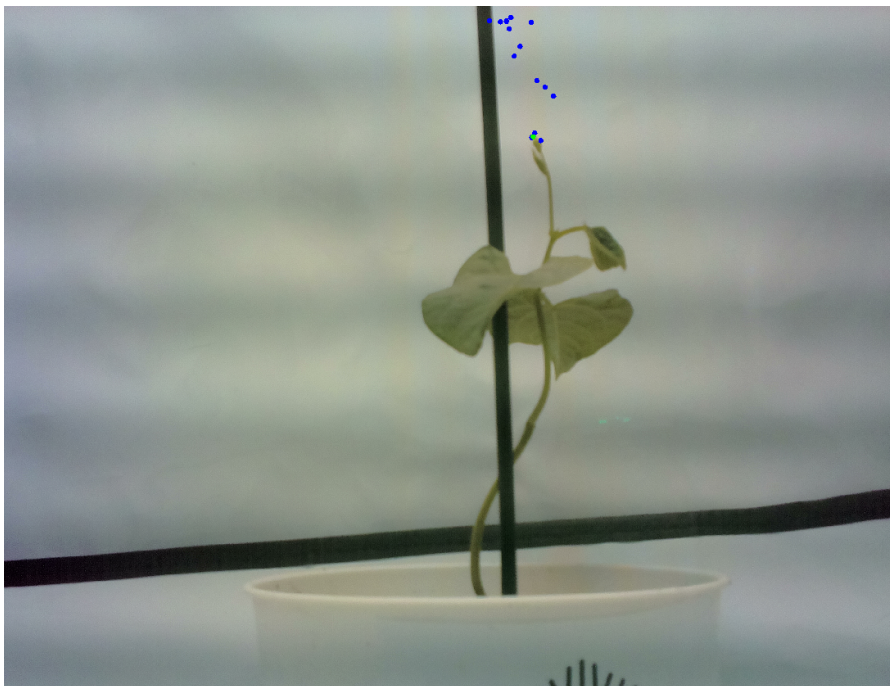


Figure 5.16: Image showing the motion tracking of tip of plant

5.4 Results

Thus Image processing was successfully utilized in deducing the tip of the plant and tracking the motion. It was seen that Superpixel algorithm was not suitable in extracting plant as it misinterpreted green regions. The Grab-cut algorithm also failed to accurately extract the plant as it mixes foreground and background pixels. The proposed algorithm works with the given setup plant images in rightly identifying the tip point. One of the drawbacks of this algorithm is that the parameters for inrange that defines boundary for green needs to be changed for every setup. This is because the intensity of plants green in image varies from one setup to another depending on the light conditions and orientation of camera. Motion tracking of the tip of the plant was also done by the use of the proposed algorithm.

CHAPTER 6

Conclusion

Thus various experiments were conducted as a part of this project which reflects different aspects of the ongoing work in florarobotica.

The garden automation provided an isolated, remote and accurate system for nurturing plants. This will consecutively ensure the growth of healthy plants at the right time which are important factors in building bio-hybrid systems and structures out of plants. In this experiment automation was carried out using RF remotes which provided a cheap, safe, simple and a reliable solution to control the day night cycle, moisture control and heating conditions.

The Plant automation experiment provided a simulation of building artefacts that mimic plant growth on a 2D plane. This formed the basis for building 3-D structural artefacts. A prototype board was built for interfacing light sensor strips. These strips would be plated using thymios automation to build braided structures. The climber plant experiments conducted in minibox setup gave insights into the nature of growth of such plants around threaded pole rods. It was observed that the pole climbers wind tightly around these rods and almost get stuck to them. The entire winding takes place in a span of one week. The processes running in the node could be reliably monitored by running them from upstart. A flash drive program was executed that ensured timely transfer of pictures from pi to flash drive. The data collected from various sensors from the node were plotted remotely and monitored through internet of things. It could be concluded from the rod movement experiment that light was of more priority than a rod during the initial stages of climber growth.

The pictures collected from mini-box experiment were analysed even further using image processing techniques. An Algorithm was proposed to deduce the tip of the plant. Further this motion of the tip was tracked. Thus the useful interpretations made from the climber experiments as discussed in chapter three and chapter four, gave an understanding of the behaviour of climber plants which would play a vital role in building the next iterations of bio-hybrid systems of plants and robotic nodes and effecting mutual synergies between them.

APPENDIX A

Appendix

A.1 Upstart

```
description "A test job file for experimenting with Upstart"
author "subha"

start on runlevel [2345]
stop on runlevel [!2345]

respawn

pre-start script
        exec echo Test Job ran at  `date` >> /var/log/testjob.log
end script

chdir /home/pi/scripts
script
        exec python cam_capture.py
end script
```

A.2 Flash Drive Transfer

```
#-------------------------------------------------------------
#Program to transfer pictures from pi to flash drive
#-------------------------------------------------------------
logging.basicConfig(filename='Transfer_output',level=logging.INFO)
NET_DRIVE='//192.168.178.1/fritz.nas/WD-Elements10B8-01/pictures/'
LOCAL_PATH='/home/pi/net/'
BACKUPFLASHDRIVE='/home/pi/BACKUP/'
store='/home/pi/store/pictures/'

def mount():
  if not (os.path.ismount('/home/pi/store/')):
```

```
    print('mounting')
    subprocess.call(['./sharingcode2.sh'])

def foldercheck1(d):
  if os.path.exists(os.path.join(store,d)):
    return
  else:
    os.makedirs(os.path.join(store,d))

def foldercheck2(d):
  if os.path.exists(os.path.join(BACKUPFLASHDRIVE,d)):
    return
  else:
    os.makedirs(os.path.join(BACKUPFLASHDRIVE,d))

#Checks Mounting of Drive
def net_check(d,f):
  if (os.path.ismount('/home/pi/store/')):
    print('mounted')
    foldercheck1(d)
    shutil.move(os.path.join(LOCAL_PATH, os.path.join(d,f)) ,
                          os.path.join(store, os.path.join(d,f)))
  else:
    print('NOT MOUNTED FAILED TO MOVE TO FLASH DRIVE')
    foldercheck2(d)
    shutil.move(os.path.join(LOCAL_PATH,os.path.join(d,f)) ,
                  os.path.join(BACKUPFLASHDRIVE, os.path.join(d,f)))
    print('moved to backup')

#Check if there are pictures in Backup Drive and transfers them first
def backup_check():
  print('BACKUP TRANSFER')
  if (os.path.ismount('/home/pi/store/')):

    for k in os.listdir(BACKUPFLASHDRIVE):
      while(os.listdir(os.path.join(BACKUPFLASHDRIVE,k))):
          for f in os.listdir(os.path.join(BACKUPFLASHDRIVE,k)):
            foldercheck1(k)
            foldercheck2(k)
            f1=os.path.join(k,f)
            shutil.move(os.path.join(BACKUPFLASHDRIVE,f1),
                                          os.path.join(store,f1))
            print('transferred file')
    print('BACKUP TRANSFER COMPLETE')
```

```
    logging.info('BACKUP TRANSFER COMPLETE')
  else:
    return

#function that transfers pictures from pi to flash drive
def transfer():
  backup_check()
  print('TRANSFER')
  logging.info('TRANSFER COMPLETE')

  for d in os.listdir(LOCAL_PATH):
    while(os.listdir(os.path.join(LOCAL_PATH,d))):
      for d1 in (os.listdir(os.path.join(LOCAL_PATH,d))):
          print "In directory %s and transferring file  %s" %(d,d1)
          net_check(d,d1)
          print('transferred  file')

# main function
if __name__ == '__main__':
  mount()
  transfer()
  print('transferred a directory')

  # Using AP Scheduler for scheduling the transfer function
  try:
    print('next schedule')
    sched.add_interval_job(mount, hours=1)
    sched.add_interval_job(transfer, minutes=2)
    sched.start()
    sys.stdout.write("Running")

  except (KeyboardInterrupt, SystemExit):
   print "Terminated"
   sched.shutdown(wait=False,shutdown_threadpool=False)
```

A.3 Plant Tip Detection

```
#-----------------------------------------------------------------
#code snippet to detect the tip of a plant
#-----------------------------------------------------------------
import numpy as np
import matplotlib
import cv2
```

```
from matplotlib import pyplot as plt

#Converting to Grey Scale
upstate_hsv = cv2.cvtColor(upstate, cv2.COLOR_BGR2HSV)

#Fixing the range for green
blue_min=np.array([17,70,100], np.uint8)
blue_max=np.array([70,255,255], np.uint8)

# get mask of pixels that are in green range
mask_inverse = cv2.inRange(upstate_hsv, blue_min, blue_max)
mask_final=cv2.cvtColor(mask_inverse, cv2.COLOR_GRAY2RGB)

# inverse mask to get parts that are not green
mask = cv2.bitwise_not(mask_inverse)

#Applying bilateral Filter
blurred = cv2.bilateralFilter(mask_inverse,5,20,20)

#Canny Edge Detection
auto = cv2.Canny(blurred,1,35)

#Adaptive Thresholding
thresh = cv2.adaptiveThreshold(auto, 255,
    cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY_INV, 11, 1)
kernel = np.ones((3, 3), np.uint8)

#Morphological Transformation
closing = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE,
                                    kernel, iterations=4)
img2,contours, hierarchy=cv2.findContours(closing.copy(),
                          cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

for c in contours:
#Applying approximate polydp
        peri = cv2.arcLength(c, True)
        app= cv2.approxPolyDP(c, 0.001 * peri, True)
        cv2.drawContours(closing, [c], -1, (0, 255, 0), 3)

new_img= a+'MODIFIED.jpg'
cv2.imwrite(new_img,closing)

img = cv2.imread(new_img)
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

```
# C. Tomasi Corner detection
corners = cv2.goodFeaturesToTrack(gray,100,0.001,7)
corners = np.int0(corners)
a=[]
b=[]

#Identifying the corner point with the least Y coordinate
for i in corners:
        x,y = i.ravel()
        a.append(y)
        b.append(x)

t=a.index((min(a)))
Y=min(a)
for l,k in enumerate(b):
    if l==t:
        cv2.circle(img,(k,Y), 9, (0,0,255), -1)#Deduction of tip point
```

Bibliography

[ASS+12] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[DSH15] Mohammad Divband Soorati and Heiko Hamann. The effect of fitness function design on performance in evolutionary robotics: The influence of a priori knowledge. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 153–160. ACM, 2015.

[HWS+15] Heiko Hamann, Mostafa Wahby, Thomas Schmickl, Payam Zahadat, Kasper Stoy, Sebastian Risi, Andres Faina, Frank Veenstra, Serge Kernbach, Igor Kuksin, Olga Kernbach, Phil Ayres, and Przemyslaw Wojtaszek. flora robotica Mixed Societies of Symbiotic Robot-Plant Bio-Hybrids. In *IEEE Symposium Series on Computational Intelligence*, pages 1102–1109, December 2015.

[RKB04] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, volume 23, pages 309–314. ACM, 2004.

[TK91] Carlo Tomasi and Takeo Kanade. *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon Univ. Pittsburgh, 1991.

[WSMH15] Mostafa Wahby, Mohammad Divband Soorati1, Sebastian Mammen, and Heiko Hamann. Evolution of Controllers for Robot-Plant Bio-Hybdrids: A Simple Case Study Using a Model of Plant Growth and Motion. In *Proc. of the 25th Workshop on Computational Intelligenc*, pages 67–86, Dortmund, Germany, December 2015.

[ZHS15] Payam Zahadat, Heiko Hamann, and Thomas Schmickl. Evolving diverse collective behaviors independent of swarm density. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 1245–1246. ACM, 2015.