

University of Ulster

School of Engineering

MEng Hons Electronics & Software

Final Project Dissertation

Title: Spider Robot using a Raspberry Pi

Author: Jebin Jose

Supervisor: Prof. Ian McCrum

Year: 2012/2013

Abstract

The purpose of this project is to control 24 radio controlled servos with an interface board for the Raspberry Pi, which will have its own microprocessors and involves developing microprocessor hardware and software to a tight real time requirement, which will enable the spider robot to move in any direction. Controlling radio-controlled servos from a raspberry is limited to one PWM output, which is of no use in controlling 24 servos. But the Raspberry pi has few compatible hardware interfaces like I2C, SPI etc. making it easy to talk to other microprocessors. Using this feature of the raspberry pi it would be possible to control servos. Using the I2C data link with the interface board has enabled to control the 24 radio-controlled servos, making the spider robot move in any direction. This will enable raspberry pi to be used for more robotic applications and cut down the cost for building a robot.

Acknowledgement

I would express all thanks to God Almighty for letting me successfully complete this project, I would also like to express my sincere gratitude to my supervisor Mr. Ian McCrum who has been my encouragement and guidance from the initial stage to the completion of the project. His support throughout the project has helped me understand the project deeply. I would also like to thank all my lecturers who have contributed to my knowledge for carrying out this project.

I would also thank my father Mr. Jose Thomas and my loving mother Mrs. Lisa Jose for their support both morally and financially. I will like to thank my siblings Mr Jaimee Jose and Mr Jaison Jose and my friends Mr. Akhil Martin and especially Ms Sheethal Avugin for the support from the beginning up to the completion of my project.

Finally I want to thank all those who have supported me in any respect for the completion of my project.

Jebin Jose

Declaration

I hereby declare that I am the sole author of this thesis. I authorize University of Ulster to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I also authorize University of Ulster to reproduce this thesis by photocopying or by any other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Signature

Date:-

Table of Contents

List of Figures	viii
Chapter 1	1
1. Introduction	1
1.1 Background/Motivation	1
1.2 Aims & Objectives	1
1.2.1 Physical Connections.....	2
1.3 Technology	2
1.4 Organization	3
1.4.1 Chapter 1:-Introduction.....	3
1.4.2 Chapter 2: - Literature Review & Background theory.....	3
1.4.3 Chapter 3: - Design & Project Planning.....	3
1.4.4 Chapter 4: -Hardware/ Software Implementation	3
1.4.5 Chapter 5: - Results	3
1.4.6 Chapter 6: - Discussion / Conclusions / Future Work.....	3
Chapter 2	4
2. Literature Review & Background theory	4
2.1 Introduction.....	4
2.2 “Chopsticks” the Spider Robot.....	4
2.3 On-going developments and research.....	5
2.3.1 PhantomX with Raspberry Pi.....	5
2.4 Raspberry Pi	7
2.5 Servos	8
2.6 Python	12
2.7 Pulse-width modulation	12
2.8 Polymorph	13
2.9 Interface Board	13
2.10 Battery.....	14
2.11 Wireless technology	14
Chapter 3	15
3. Design & Project Planning.....	15
3.1 Concept design	15

3.2	Final Design.....	17
3.3	Components	18
3.4	Walking Sequence.....	19
3.4.1	Zigzag walking method.....	20
3.4.1.1	Forward Zigzag walking	20
3.4.1.2	Backward Zigzag walking	20
3.4.1.3	Turning Left	21
3.4.1.4	Turning Right	21
3.4.2	Alternative walking method.....	22
3.4.2.1	Forward Alternative walking	22
3.4.2.2	Backward Alternative walking	22
3.4.2.3	Turning Left	23
3.4.2.4	Turning Right	23
3.5	Webcam Mounting Design	24
3.6	Web Interface Sketching.....	25
Chapter 4	26
4.	Hardware/Software Implementation	26
4.1	System Design	26
4.2	Hardware Implementation.....	27
4.2.1	Building the Spider Robot Skeleton	27
4.2.2	Connections.....	30
4.2.3	Testing Servos.....	31
4.3	Software Implementation	32
4.3.1	Setting up Raspberry Pi.....	32
4.3.1.1	Installing Raspbian “wheezy”	33
4.3.1.2	Installing VNC Server	33
4.3.1.3	Wi-Fi connection.....	34
4.3.2	Setting up Interface board	35
4.3.3	Setting up Webcam	36
4.3.3.2	Setting up Webcam Server	37
4.3.4	Coding the Servos	38
4.3.4.1	Frequency/pulse length	38
4.3.4.2	Initializing PWM devices	39
4.3.4.3	Coding the spider robot functions	39
4.3.5	Web interface	42
4.3.6	Server start-up.....	43
4.4	Implemented System	43

4.5	Flow chart	44
4.5.1	Server flowchart.....	44
4.5.2	Client Flowchart.....	44
Chapter 5	46
5.	Results	46
5.1	Unit Test	46
5.2	Web Interface test results.....	47
5.3	Spider robot	48
Chapter 6:	50
6.	Discussion / Conclusions / Future Work	50
6.1	Discussion.....	50
6.2	Conclusions	50
6.3	Future Work.....	51
Reference	52
Bibliography	54
Appendix A.....		55
Zigzag Walking Method.....		55
Alternative Walking Method.....		63
Appendix B.....		71
Appendix C.....		73
Appendix D		74
Appendix E.....		84

List of Figures

<i>Figure 2-1: The Chopstick Spider Robot.....</i>	<i>6</i>
<i>Figure 2-2: PhantomX Spiderbot.....</i>	<i>8</i>
<i>Figure 2-3: Raspberry Pi.....</i>	<i>10</i>
<i>Figure 2-4: Servo.....</i>	<i>11</i>
<i>Figure 2-5: A Servo Disassembled.....</i>	<i>12</i>
<i>Figure 2-6: The Pulse Width Modulation of a Servo</i>	<i>13</i>
<i>Figure 2-7: Raspberry Pi GPIO ports.....</i>	<i>16</i>
<i>Figure 3-1: First design using a raspberry pi and PIC16F877.....</i>	<i>18</i>
<i>Figure 3-2: Final design of the Spider robot.....</i>	<i>19</i>
<i>Figure 3-3: One leg forward movement of a spider robot.....</i>	<i>21</i>
<i>Figure 3-4: Forward Zigzag walking.....</i>	<i>22</i>
<i>Figure 3-5: Backward Zigzag walking.....</i>	<i>22</i>
<i>Figure 3-6: Turning Left using Zigzag method</i>	<i>23</i>
<i>Figure 3-7: Turning Right using Zigzag method</i>	<i>23</i>
<i>Figure 3-8: Forward Alternative walking</i>	<i>24</i>
<i>Figure 3-9: Backward Alternative walking</i>	<i>24</i>
<i>Figure 3-10: Turing Left using Alternative method.....</i>	<i>25</i>
<i>Figure 3-11: Turning Right using Alternative method</i>	<i>25</i>
<i>Figure 3-12: Webcam mounting kit design</i>	<i>26</i>
<i>Figure 3-13: Webcam mounting kit.....</i>	<i>26</i>
<i>Figure 3-14: Web Interface design</i>	<i>26</i>
<i>Figure 4-1: System Design.....</i>	<i>26</i>
<i>Figure 4-2: Base model of the spider robot.....</i>	<i>38</i>
<i>Figure 4-3: Modified base of the spider robot.....</i>	<i>38</i>
<i>Figure 4-4: Top Structure of spider robot</i>	<i>29</i>
<i>Figure 4-5: Finished model of the spider robot.....</i>	<i>29</i>
<i>Figure 4-6: Modified Leg.....</i>	<i>30</i>
<i>Figure 4-7: Connections</i>	<i>30</i>
<i>Figure 4-8: Testing servos with Turnigy Servo Tester</i>	<i>31</i>
<i>Figure 4-9: Rasbian installed on Raspberry pi.....</i>	<i>33</i>

<i>Figure 4-10: VNC Display Session</i>	<i>34</i>
<i>Figure 4-11: Modules.....</i>	<i>35</i>
<i>Figure 4-12: Shows the I2C device found.....</i>	<i>36</i>
<i>Figure 4-13: Intial Web Interface</i>	<i>42</i>
<i>Figure 4-14: Overall System</i>	<i>43</i>
<i>Figure 4-15: server flowchart.....</i>	<i>44</i>
<i>Figure 4-16: client Flow chart.....</i>	<i>45</i>
<i>Figure 5-1: Client system test result.....</i>	<i>47</i>
<i>Figure 5-2: Spider robot top view.....</i>	<i>48</i>
<i>Figure 5-3: Spider robot front view.....</i>	<i>49</i>

Chapter 1

1. Introduction

The introduction gives the outline of scope and context of the project.

Project title: Spider Robot using Raspberry Pi

Prof. Ian McCrum, who is a lecturer in engineering faculty, did propose this project. The successful development of this project will enable Raspberry Pi to control 24 Servos using an Interface board. The project is done by connecting a raspberry pi to the interface board with an embedded computer that has the capabilities to control 24 radio-controlled servos. It also consists of an extra battery to power the raspberry pi, microcontroller, servos and also a voltage regulator to give the exact power to the raspberry pi. Once all the physical connection are done, the next main part is the programming section. The programing of the raspberry pi is done in python.

1.1 Background/Motivation

This project is based on controlling radio-controlled servos using a raspberry pi. An interface board will be mounted onto the raspberry pi, which will directly control the radio-controlled servos using the signals from the raspberry pi.

1.2 Aims & Objectives

To develop a eight legged spider robot that can be driven using servos and can be controlled by a mobile devices over the Wi-Fi.

The core objectives are:

- Gather system requirements
- Evaluate and study the platform required for the system

- Evaluate and study suitable development language, technologies and tools
- Evaluate Methods of Interface
- Program Raspberry Pi
- Program Interface board for servos
- Program Mobile control app
- Evaluate and test the system
- Maintain system

1.2.1 Physical Connections

- The connection made from the Raspberry pi to the voltage regulator, this is done so as to connect the Raspberry pi to an external battery.
- The connection made from the interface board to the voltage regulator, this is done so as to connect the interface board to another external battery.
- The connection of an external battery to the Raspberry pi.
- The connection of an external battery to the interface board.
- The Servos to the interface board.
- The fixation of all these components on the spider robot frame.

1.3 Technology

The technology used in the project is python as the libraries available are only in python. Also it allows creating a user interface so that the user can see and control certain movements of the robot. We use the wireless technology to transmit data of the raspberry pi to the users system.

1.4 Organization

This project is been divided into five chapters mainly as follows:

1.4.1 Chapter 1:-Introduction

In this section it gives a general idea of the project and also about the Aims and objectives of this project.

1.4.2 Chapter 2: - Literature Review & Background theory

This section contains information of the raspberry pi and the chopstick spider robot, also the previous and on-going researches and developments regarding these components in different fields.

1.4.3 Chapter 3: - Design & Project Planning

This chapter contains information about the design of the spider robot and the project planning

1.4.4 Chapter 4: -Hardware/ Software Implementation

In this the hardware tools and the software tools required for the carrying out of the project is been specified also the interface designs for the project are also discussed.

1.4.5 Chapter 5: - Results

This chapter contains the test results of what has been done and what more of the objectives could have been achieved.

1.4.6 Chapter 6: - Discussion / Conclusions / Future Work

This chapter contains the discussion, conclusion and the future work of the project.

Chapter 2

2. Literature Review & Background theory

2.1 Introduction

This section of the report consists of all past, presently on-going and also upcoming researches or developments in area of raspberry pi driven spider robot. Also in this section we will talk about both the raspberry pi and servos. Raspberry pi is a great small computer, which can do a lot of things and have different applications in the field of robotics. In the following part some of the areas of application of the raspberry pi and the chopsticks spider robot will be covered. The first sections will be about the chopsticks spider robot.

2.2 “Chopsticks” the Spider Robot

The chopstick spider robot is a multi- limbed robot that is created with disposable bamboo chopsticks, Polymorph and an Arduino-compatible controller as shown as in Figure 2-1. This multi- limbed robot is a relatively cheap and is a simple 8-legged robot. The robot is constructed using chopsticks and is stuck together into shape using polymorph. The polymorph is heated to a temperature around 60 degrees and then mounted into the desired shape. The spider robot runs on a Red Back Spider controller which is an Arduino compatible robot controller designed specifically for robots that use a large number of servos.

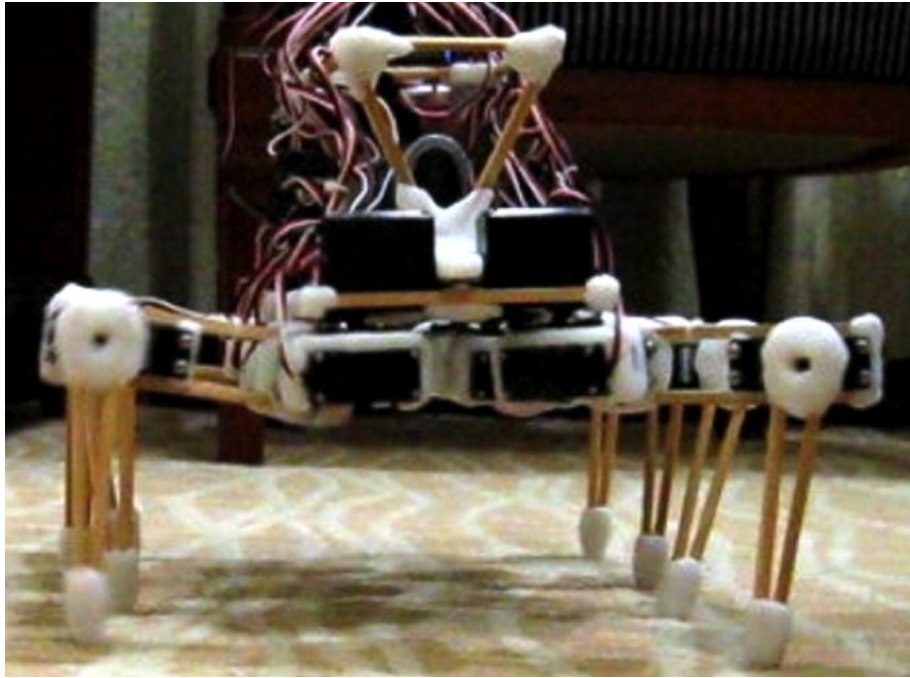


Figure 2-1: The Chopstick Spider Robot

The Spider robot made up of 8 legs has 3 servos for each leg which makes it a total of 24 servos for the 8 legs. It is also fitted with two IR compound eyes for close range tracking on a pan/tilt kit.

2.3 On-going developments and research

There are number of developments and research's going on with the raspberry pi due to the fact its cheap and very small compared to other computer in the market now. But there is one-spider robot build with a raspberry pi and that to in development. It is called the phantomX with raspberry pi.

2.3.1 PhantomX with Raspberry Pi

PhantomX is a Hexapod robot kit which combines some of the most advanced robotic technologies on the market to bring unparalleled performance and quality into your hands. The phantomX which is the PhantomX AX Hexapod Mark II was

created by Interbotix Labs. The phantomX Kit comes with everything you need for a fully featured robotic platform. All 18 DYNAMIXELs frame components, anodized black socket-head hardware, an Arduino-compatible ArbotiX Robocontroller, FTDI interface for programming, the handheld ArbotiX Commander, a set of paired Xbees, a PC-side Xbee USB interface, a 3S 2200mAh LiPo Battery, Multi-Function LiPo Balance Charger [101]. The phantomX with raspberry pi is a project by Kevin Ochs who used the raspberry pi to drive the phantom skeleton as shown as in figure below.



Figure 2-2: PhantomX Spiderbot

The project contained of a Raspberry Pi, which was stuck onto a inanimate frame of a six-legged robot, giving it a robotic body life.

The spiderbot, runs of a Phoenix software customized in C++. The PhantomX also runs a host of other softwares, codes to power the servos is running a host of other software as well, from code to power the servos and to providing the user with remote control.

2.4 Raspberry Pi

The Raspberry Pi is a single-board computer the size of a credit-card-sized developed in the UK by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools. The Raspberry Pi is manufactured through licensed manufacturing deals with Element 14/Premier Farnell and RS Components. Both of these companies sell the Raspberry Pi online. The Raspberry Pi has a Broadcom BCM2835 system on a chip (SoC), which includes an ARM1176JZF-S 700 MHZ processor (The firmware includes a number of "Turbo" modes so that the user can attempt overclocking, up-to 1 GHz, without affecting the warranty), VideoCore IV GPU, and originally shipped with 256 megabytes of RAM, later upgraded to 512MB. It does not include a built-in hard disk or solid-state drive, but uses an SD card for booting and long-term storage.

The pi can be installed with Debian and Arch Linux ARM which can be downloaded from the raspberry pi website. Also planned are tools for supporting Python as the main programming language, with support for BBC BASIC, C, and Perl [1].

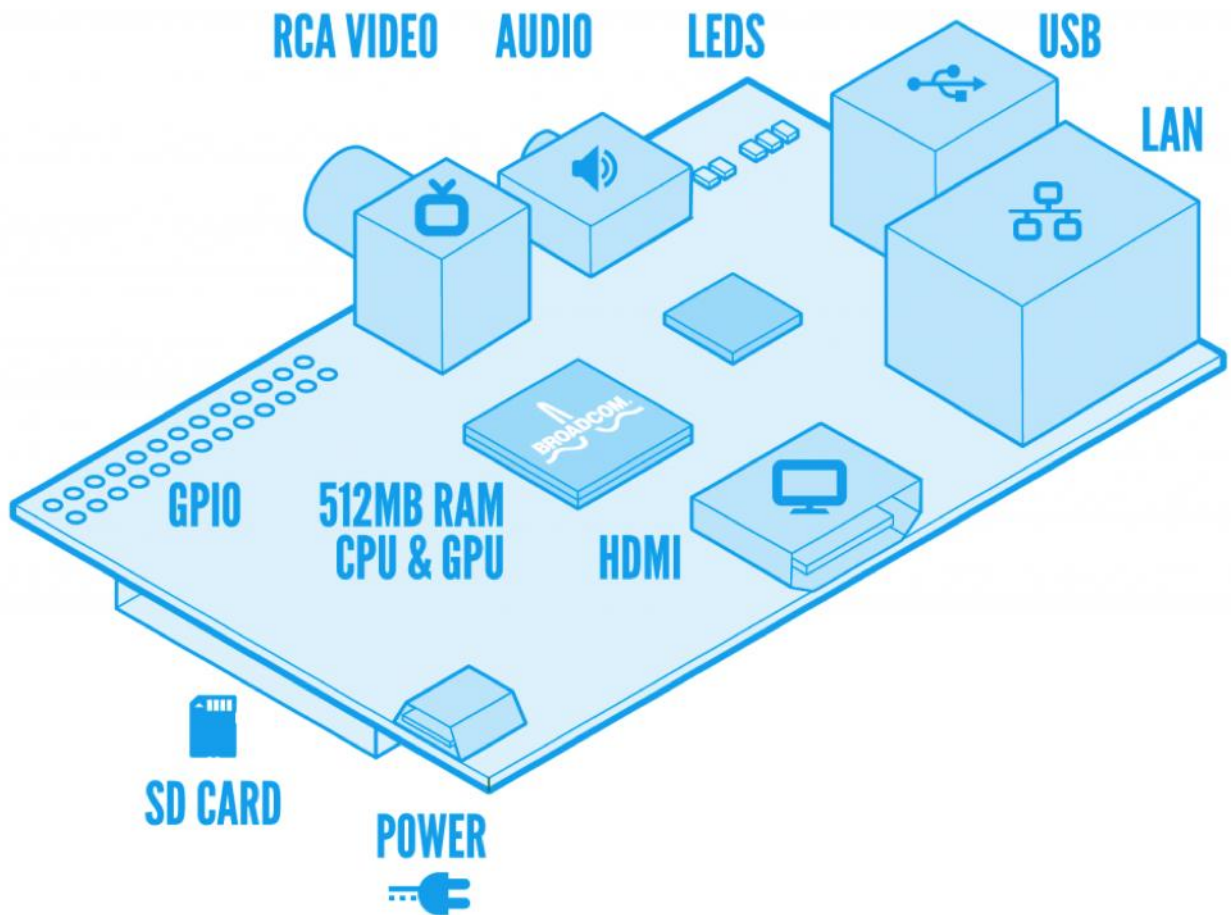


Figure 2-3: Raspberry Pi

2.5 Servos

A Servo is a small device that has an output shaft. This shaft can be positioned to specific angular positions by sending the servo a coded signal. As long as the coded signal exists on the input line, the servo will maintain the angular position of the shaft. As the coded signal changes, the angular position of the shaft changes. In practice, servos are used in radio-controlled airplanes to position control surfaces like the elevators and rudders.

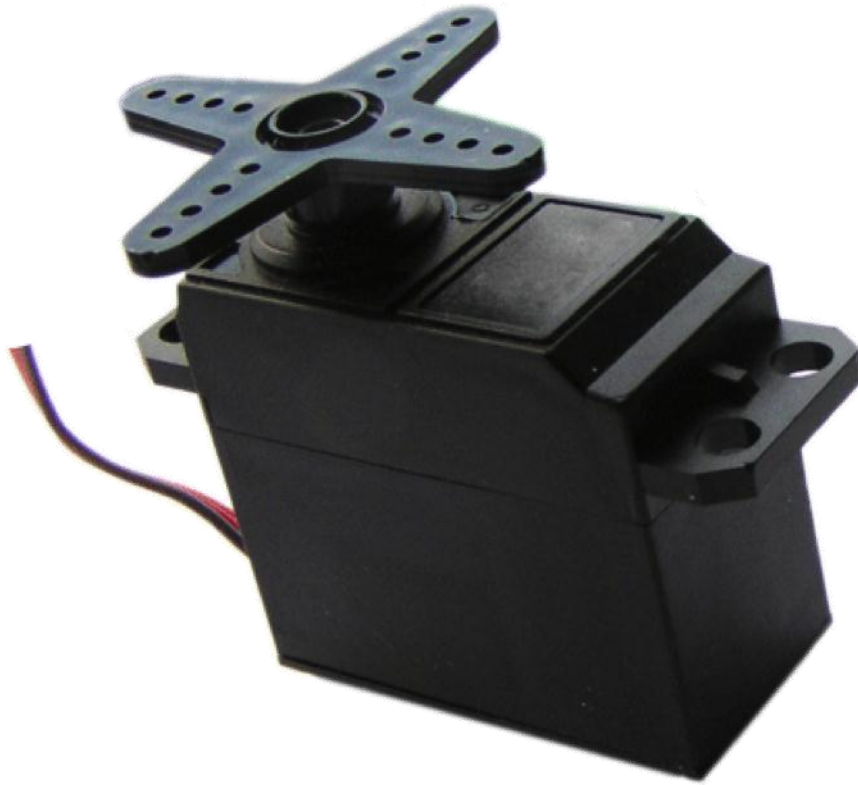


Figure 2-4: Servo

Servos are extremely useful in robotics. The motors are small, as you can see by the picture above, have built in control circuitry, and are extremely powerful for their size. A standard servo has 42 oz/inches of torque, which is pretty strong for its size. It also draws power proportional to the mechanical load. A lightly loaded servo, therefore, doesn't consume much energy. The guts of a servomotor are shown in the figure 2-5 below. You can see the control circuitry, the motor, a set of gears, and the case. You can also see the 3 wires that connect to the outside world. One is for power (+5volts), ground, and the white wire is the control wire.

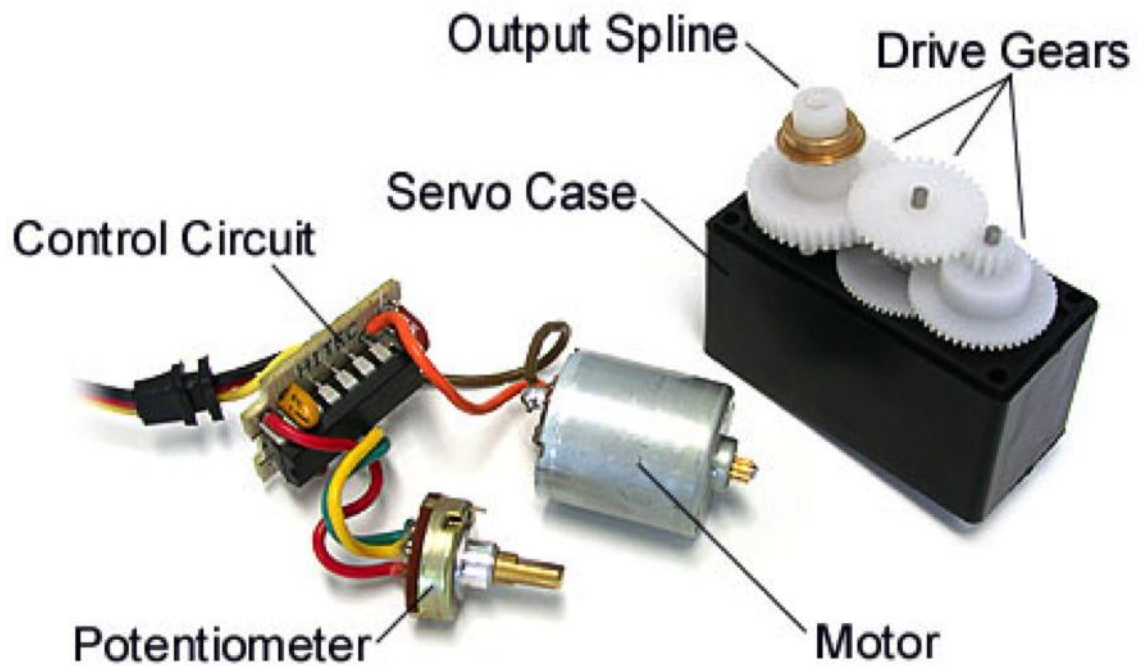


Figure 2-5: A Servo Disassembled

The servo motor has some control circuits and a potentiometer (a variable resistor, aka pot) that is connected to the output shaft. In the figure 2-5 above, the pot can be seen on the right side of the circuit board. This pot allows the control circuitry to monitor the current angle of the servo motor. If the shaft is at the correct angle, then the motor shuts off. If the circuit finds that the angle is not correct, it will turn the motor the correct direction until the angle is correct. The output shaft of the servo is capable of travelling somewhere around 180 degrees. A normal servo is used to control an angular motion of between 0 and 180 degrees. A normal servo is mechanically not capable of turning any farther due to a mechanical stop built on to the main output gear.

The amount of power applied to the motor is proportional to the distance it needs to travel. So, if the shaft needs to turn a large distance, the motor will run at full speed. If it needs to turn only a small amount, the motor will run at a slower speed. This is called proportional control. The control wire is used to communicate the angle. The angle is determined by the duration of a pulse that is applied to the control wire. This is called Pulse Coded Modulation. The servo expects to see a pulse every 20 milliseconds (.02 seconds). The length of the pulse will determine

how far the motor turns. A 1.5 millisecond pulse, for example, will make the motor turn to the 90 degree position (often called the neutral position). If the pulse is shorter than 1.5 ms, then the motor will turn the shaft to closer to 0 degrees. If the pulse is longer than 1.5ms, the shaft turns closer to 180 degrees.

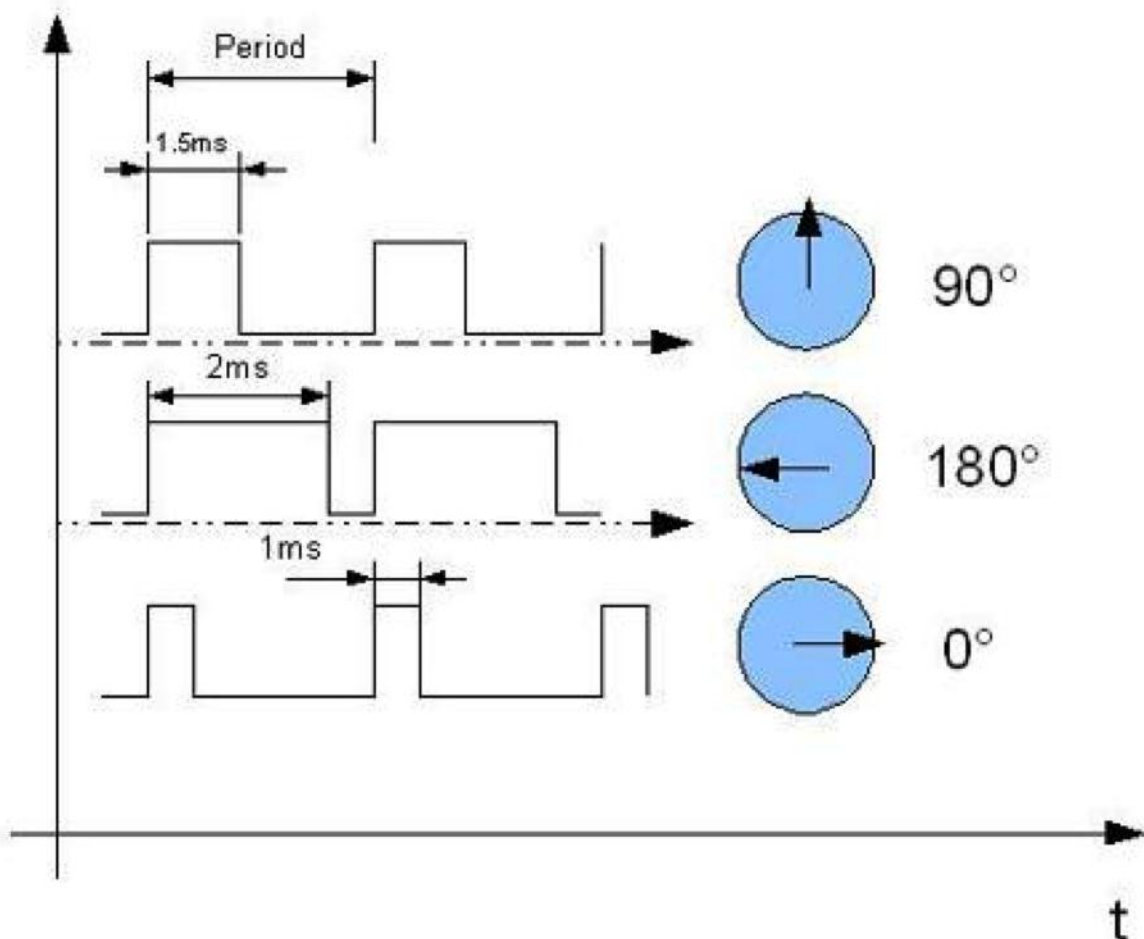


Figure 2-6: The Pulse Width Modulation of a Servo

As you can see in the figure 2-6, the duration of the pulse dictates the angle of the output shaft (shown as the green circle with the arrow).

2.6 Python

Python is a general-purpose, high-level programming language whose design philosophy emphasizes code readability. Python's syntax allows for programmers to express concepts in fewer lines of code than would be possible in languages such as C, and the language provides constructs intended to enable clear programs on both a small and large scale.

Python supports multiple programming paradigms, including object-oriented, imperative and functional programming styles. It features a fully dynamic type system and automatic memory management, similar to that of Scheme, Ruby, Perl and Tclm and has a large and comprehensive standard library [6].

2.7 Pulse-width modulation

Pulse-width modulation (PWM), or pulse-duration modulation (PDM), is a commonly used technique for controlling power to inertial electrical devices, made practical by modern electronic power switches.

The average value of voltage (and current) fed to the load is controlled by turning the switch between supply and load on and off at a fast pace. The longer the switch is on compared to the off periods, the higher the power supplied to the load is.

The PWM switching frequency has to be much faster than what would affect the load, which is to say the device that uses the power.

The term duty cycle describes the proportion of 'on' time to the regular interval or 'period' of time; a low duty cycle corresponds to low power, because the power is off for most of the time. Duty cycle is expressed in percent, 100% being fully on.

The main advantage of PWM is that power loss in the switching devices is very low. When a switch is off there is practically no current, and when it is on, there is almost no voltage drop across the switch. Power loss, being the product of voltage and current, is thus in both cases close to zero. PWM also works well with digital controls, which, because of their on/off nature, can easily set the needed duty cycle [5].

2.8 Polymorph

Polymorph is one of a new generation of commercial plastics with unusual properties. It is one of a class of polymers, which have remarkably low fusing temperatures of around 60°. In other words, Polymorph can be reduced to a moldable condition by immersion in hot water.

Once it has fused and turns from opaque to clear, it can be molded in a variety of ways and stays workable until a much lower temperature. When fully cooled it is very similar in appearance and physical properties to a broad range of polythenes. It is stiffer, stronger and tougher than other plastics. It is a true thermoplastic and can thus be re-heated and thermoformed any number of times [3].

2.9 Interface Board

There are a number of ways Raspberry Pi can connect to an interface board. Some of which are General Purpose Input/Output (GPIO), Inter-Integrated Circuit (I2C), Serial Peripheral Interface Bus (SPI), Universal Asynchronous Receiver/Transmitter (UART), RS323. The most common of all the above connections are the I2C data link. The I2C data link can connect the raspberry and the PCA9685 microprocessor. The PCA9685 has 16 channel PWM output and can be addressed to another PCA9685 chip making it a 32 channel PWM output, which is ideal for this project.

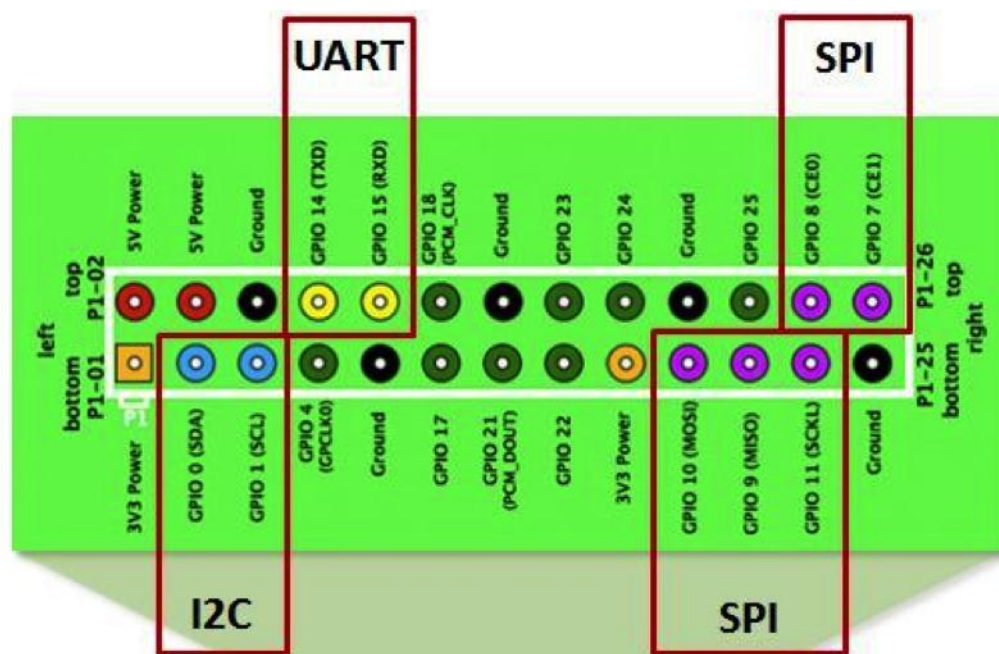


Figure 2-7: Raspberry Pi GPIO ports

2.10 Battery

Battery was an other important part of the project since the robot would be controlled wirelessly. The battery had to be powerful enough to support the servos and the raspberry pi. But one battery will cause problems in the later stage of the project since the switching directions on the servo can cause a lot of noise on the supply, and the servos will cause the voltage to fluctuate significantly, which is a bad situation for the Pi. This lead to choose two batteries for this project. The best batteries for the project was Lipo batteries because the Lithium-ion polymer batteries had a lower weight and greatly increased run times and power delivery can be sufficient enough to run 24 servos.

2.11 Wireless technology

The wireless technology involves the transfer of information between different points, which are not physically connected to each other. The distance between the points could be in the range of a few meters as in the case of a television – remote control, to millions of kilometres as in the radio communications done in free space. A WLAN (Wireless local area network) uses a wireless distribution method to link two or more devices and usually involves a connection to the internet through an access point. It enables the user to move around within the range of the WLAN without losing the connection to the network.

Chapter 3

3. Design & Project Planning

As a result of the literature review and the research done on the spider robot, raspberry pi and the servos in various fields the project is been narrowed or rather the focus was shifted from the wider range of the concept of application of raspberry pi in robotics to a smaller range considering the time frame and tools or materials available. The main disadvantage or constrain of this project was that there are very less number of articles or papers published regarding the technology. This is because the raspberry pi technology is a relatively new technology, which was released around almost one year back. This concept of the project is from the chopstick spider robot. The power of the raspberry pi to do multiple things at the same time is the main reasons for choosing this project.

Now once the research was completed and the adoption of a technology was decided the next step was to plan the project and put the knowledge found in the right direction for the completion of the project. A plan was devised for the completion of the project by attaining the objectives that are described in this section.

3.1 Concept design

Since the main part of this project was to control 24 radio controlled servos using a raspberry pi and a interface board. The skeleton of the spider robot was built using the chopsticks spider robot model shown as in figure 2-1. A detailed instruction of how to make the spider robot skeleton is in Appendix D. The first concept design was to use the raspberry pi with a PIC16F877 chip to control the servos as shown as in figure 3-1.

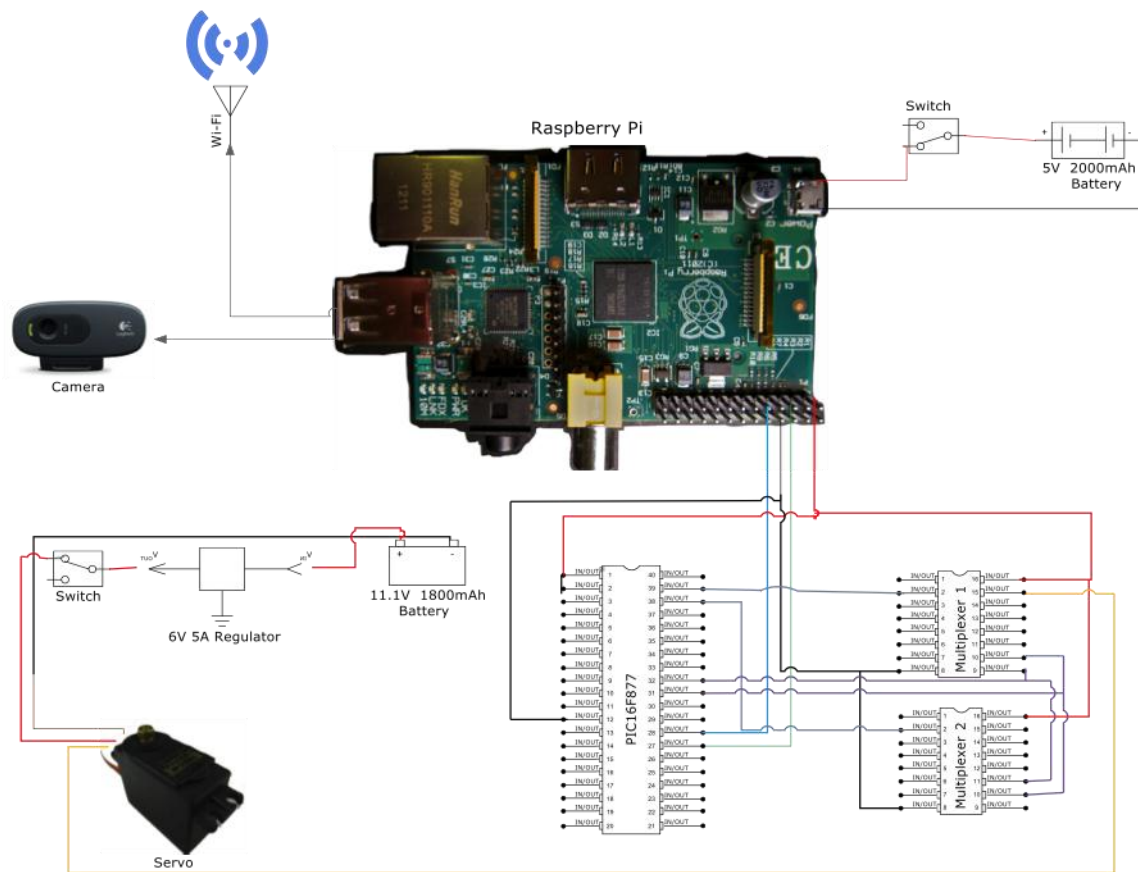


Figure 3-1: First design using a raspberry pi and PIC16F877

The figure above shows how all the connections are made; the raspberry pi is connected to a power source through a switch, the power source here is a battery of 5V 2000mah. The Wi-Fi dongle and the Logitech C200 camera are connected to the USB port of the raspberry pi. The PIC16F877 is connected to the raspberry pi through the GPIO pins of the pi. The PIC16F877 is powered from the GPIO pins of the raspberry pi and is connected to the I2C pins of the GPIO for the data connection. The PIC16F877 which has two PWM (Pulse-width modulation) outputs are connected to two multiplexer in order to multiplex it to the 24 servos. The servos have three inputs connections of which two are positive and negative (ground) volts inputs and the other is the PWM (Pulse-width modulation) input. The servos are connected to 11.1V 1800mAh battery through a switch and a 6V 5A regulator.

3.2 Final Design

From doing the further background research for all the compounds in the concept design, there was problem with the PIC16F877 chip. Since the PIC16F877 had only two PWM outputs and had to use two multiplexers to output the signal to 24 servos it can cause a large delay (spider won't walk fast enough) and also problems between the servos. After researching for another way to make the connection to the servos from the raspberry pi, found a raspberry pi expansion board for servos - I2C 32 Channel 12 bit PMW Servo Board. The I2c 32 channel PWM servo board had two PCA9685 chips, which outputted to the 32 channel PWM output. The two chips were powered from the GPIO pins of the raspberry pi and there was is a separate power input for the radio-controlled servos on the board. The I2c 32 channel PWM servo board connected directly into the GPIO pins of the raspberry pi and had the correct shape to fit above the raspberry pi making it a small unit. A fully detailed data sheet of the I2C 32 channel PWM servo board is in Appendix E.

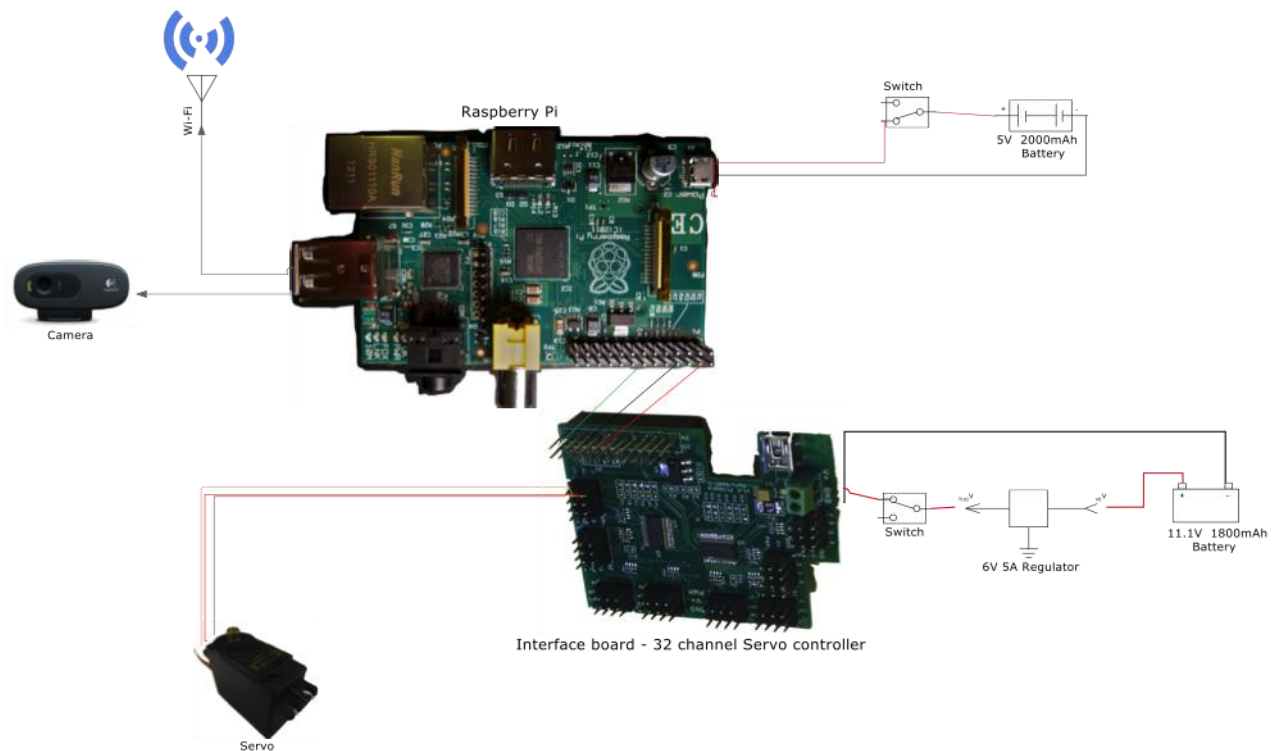


Figure 3-2: Final design of the Spider robot

The final design of the electronics as shown as in figure 3-2 is a simple design. The design is almost same as the concept design above with a few changes. The I2c 32

channel PWM servo board replaces the PIC16F877 and the multiplexers. And also the 11.1V 1800mAh battery is connected to the power input of the interface board, which supplies the 24 radio-controlled servos. In this design all the servos are plugged into the interface board and the battery is connected to the power input of the board through a 6V 5A regulator.

3.3 Components

There are several components needed for this project, the list for components are shown below: -

1. Raspberry Pi
2. USB Keyboard and Mouse
3. SD card
4. Powered USB-Hub
5. WI-FI dongle
6. HDMI Cable
7. Servos
8. Webcam
9. I2c 32 channel PWM Servo Board (1)
10. 12Kg/cm servo with metal gears (8)
11. 5Kg/cm servo (16)
12. Miniature servos (2)
13. Polymorph granules (500g)
14. Disposable Chopsticks Bamboo (40+)
15. 5V Rechargeable battery pack 2000mAh or better (1)
16. 11.1V Rechargeable battery pack 1800mAh or better (1)
17. Switch (2)
18. 6V 5A Regulator
19. Cable ties light duty
20. Self-tapping screws Small pan head 2 x 6mm (200)
21. Self-tapping screws Small pan head 2 x 8mm (50)

3.4 Walking Sequence

For a real spider to walk it has to lift its legs and place it in the direction it wants to go in. But in a spider robot each leg is made up of 3 servos and has to move all the three servos to perform the task of walking.

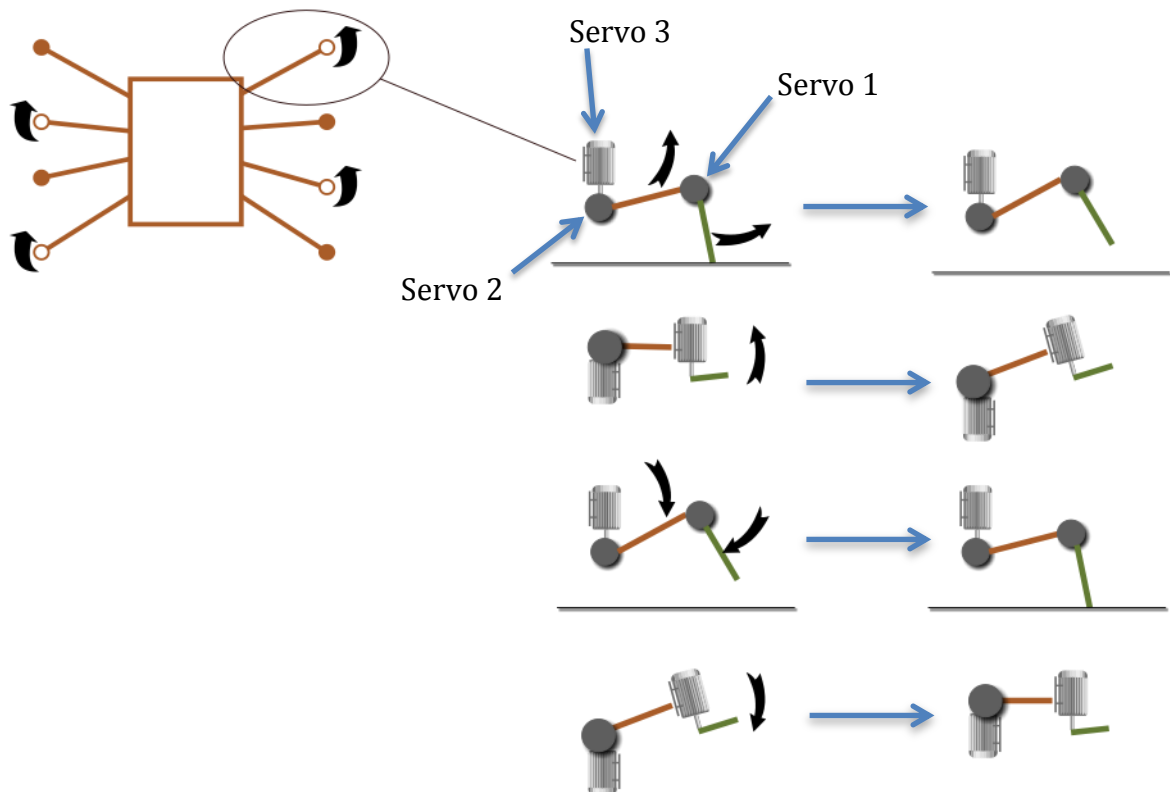


Figure 3-3: One leg forward movement of a spider robot

Figure 3-3 shows how one leg (3 servos) performs the task of walking forward. First the spider robot lifts the leg by turning servo 1 and 2 up, then servo 3 is turned to around 45 degrees forward. Finally servo 1 and 2 turns back to its original position and servo 3 turns in the opposite direction that is backwards to pull the robot forward.

There are different ways a spider can walk, out of which two of the ways were selected for this project. They are the zigzag walking and alternative walking.

3.4.1 Zigzag walking method

Zigzag walking method is a simple way of walking. In this method the zigzag legs are lifted and placed forward. For example, the 2nd, 3rd, 6th, 7th legs are lifted and placed forward when the rest are stationary as shown as in figure 3-3.

3.4.1.1 Forward Zigzag walking

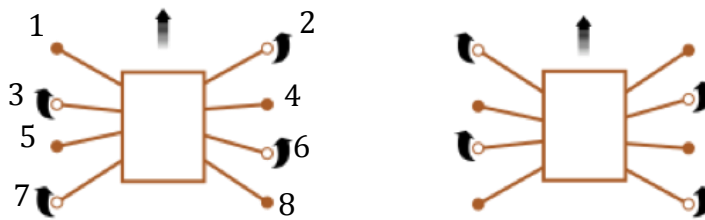


Figure 3-4: Forward Zigzag walking

Figure 3-4 shows how the forward zigzag walking is done. The 2nd, 3rd, 6th and 7th legs are lifted and placed around 45 degrees forward and brought back to its original position which moves the spider robot forward, then the 1st, 4th, 5th and 8th legs does the same tasks as the other legs to complete one cycle of walking. This is repeated to perform the task of forward walking.

3.4.1.2 Backward Zigzag walking



Figure 3-5: Backward Zigzag walking

Figure 3-5 shows how a backward zigzag walking is done. The backward zigzag walking is the same as the forward walking but this time the lifted legs are placed

45 degrees backward and brought back to its original position to give the backward motion.

3.4.1.3 Turning Left



Figure 3-6: Turning Left using Zigzag method

Figure 3-6 shows how a spider robot turns left using the zigzag method. In this task the 2nd, 6th legs move forward and the 3rd, 7th legs move backward and then brought back to its original position to make the spider robot turn left.

3.4.1.4 Turning Right



Figure 3-7: Turning Right using Zigzag method

Figure 3-6 shows how a spider robot turns right using the zigzag method. In this task the 2nd, 6th legs move backwards and the 3rd, 7th legs move forward and then brought back to its original position to make the spider robot turn right.

3.4.2 Alternative walking method

Alternative walking method is another simple way of walking. In this method the alternative legs are lifted and placed forward. For example, the 1st, 2nd, 5th and 6th legs are lifted and placed forward when the rest are stationary.

3.4.2.1 Forward Alternative walking



Figure 3-8: Forward Alternative walking

Figure 3-8 shows how the forward alternative walking is done. This time the 1st, 2nd, 5th and 6th legs are lifted and placed around 45 degrees forward and brought back to its original position which moves the spider robot forward, then the 3rd, 4th, 7th and 8th legs do the same tasks as the other legs to complete one cycle of walking. This is repeated to perform the task of forward walking.

3.4.2.2 Backward Alternative walking



Figure 3-9: Backward Alternative walking

Figure 3-9 shows how a backward alternative walking is done. The backward alternative walking is the same as the forward walking but this time the lifted legs

are placed 45 degrees backward and brought back to its original position to give the backward motion.

3.4.2.3 Turning Left



Figure 3-10: Turing Left using Alternative method

Figure 3-10 shows how a spider robot turns left using the alternative method. In this task the 2nd, 6th legs move forward and the 1st, 5th legs move backward and then brought back to its original position to make the spider robot turn left.

3.4.2.4 Turning Right



Figure 3-11: Turning Right using Alternative method

Figure 3-6 shows how a spider robot turns right using the alternative method. In this task the 2nd, 6th legs move backwards and the 1st, 5th legs move forward and then brought back to its original position to make the spider robot turn right.

3.5 Webcam Mounting Design

The webcam mounting design had two radio-controlled servos and a chopstick frame as shown as in figure 3-12.

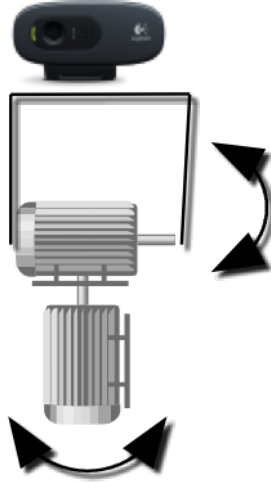


Figure 3-12: Webcam mounting kit design

The webcam is mounted onto the chopstick frame (made of chopstick and polymorph). The chopstick frame is mounted onto a radio-controlled servo, which turns the chopstick frame giving the camera top and bottom view. This radio-controlled servo is mounted onto another radio-controlled servo, which is vertical mounted onto the spider robots frame and thus giving the camera the left and right view.



Figure 3-13: Webcam mounting kit

Figure 3-13 shows the built webcam mounting kit. The chopstick frame is fixed to the servos using screws.

3.6 Web Interface Sketching

The web interface is a web page from which the spider robot is controlled. The web interface should be able to interact with the raspberry pi and call functions as buttons are pressed on the page.

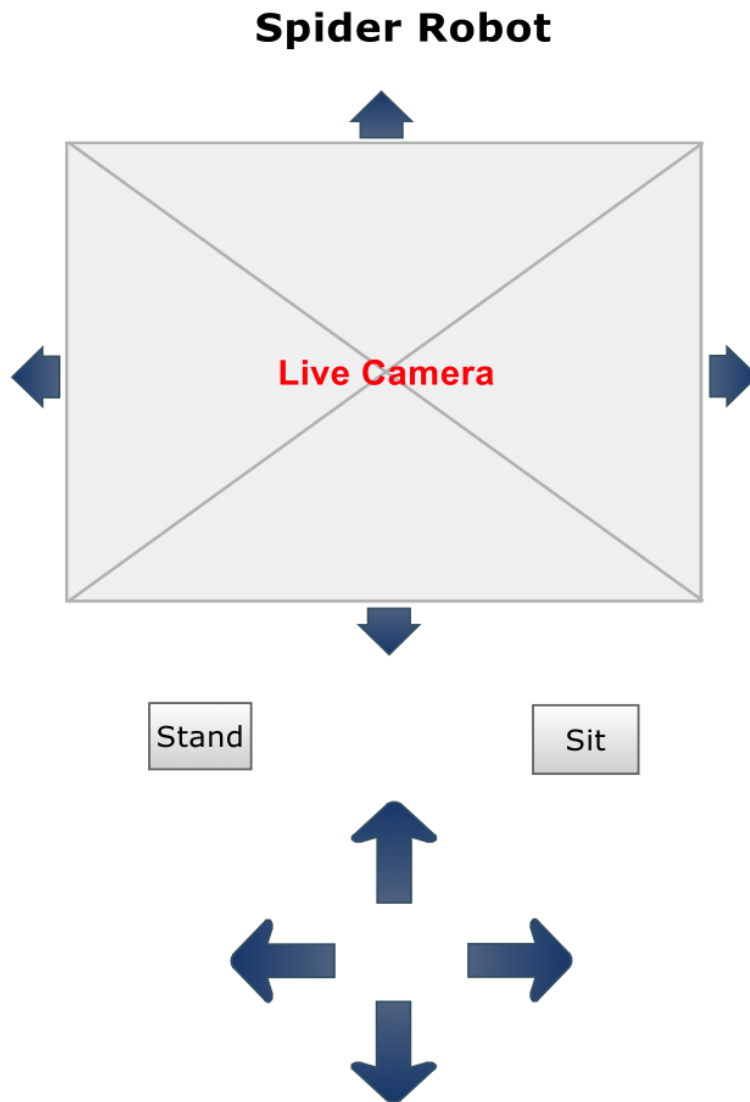


Figure 3-14: Web Interface design

Figure 3-14 shows the design of the web interface. The web interface should be able to stream the webcam on the spider robot and display it on the web page and should be able to send the commands to the raspberry pi. There are 10 main controls for the spider robot: forward, backward, left, right, sit, stand, webcam up, webcam down, webcam left and webcam right. The web interface is developed in html and JavaScript language.

Chapter 4

4. Hardware/Software Implementation

As a result of the Background research of the existing systems and different technology available a proper plan was done and a procedure for implementing the technology was set as said in the previous section of the Design and project planning.. Some of the above mentioned technologies were implemented in the project. According to the system designed for the project it involves both hardware and software tools implementation. In this section we discuss the hardware tools and software selected and implemented for the completion of the project.

4.1 System Design

The system is designed in such a way that the external system (Interface board) is only the receiving end and not an input end. But the client system receives and sends data to the main system (Raspberry Pi) and also is designed to be connected to the main system using the Wi-Fi network. The system design is as shown in the below figure 3-1.

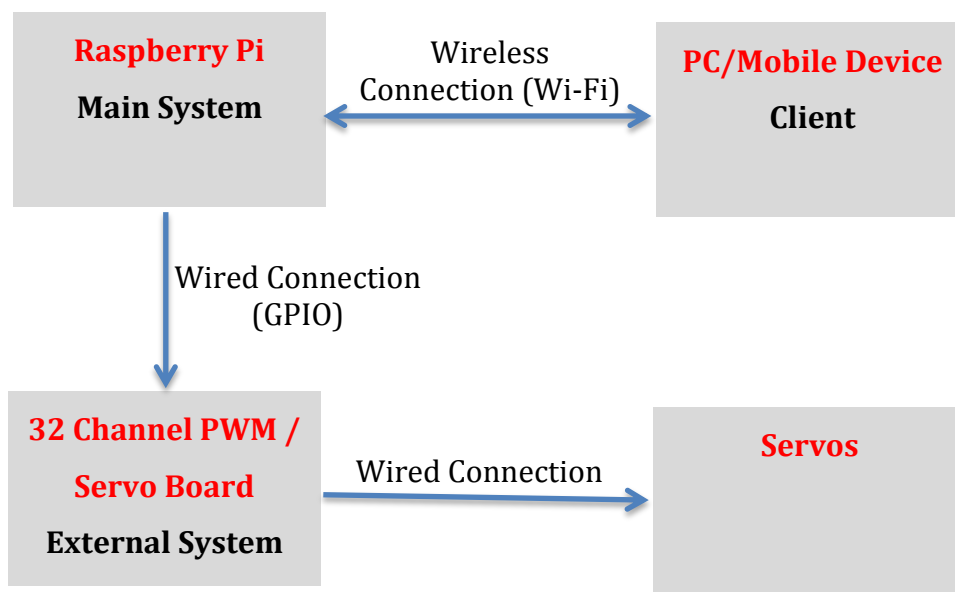


Figure 4-1: System Design

4.2 Hardware Implementation

After a thorough investigation of technologies and hardware tools that could be used for the proposed project it was decided to use a certain set of the hardware tools and these tools are mentioned in this section. The hardware tools required for getting the standard of the proposed project are as listed below

- Raspberry Pi: - It is the device where the whole project revolves around and this device is a small computer with a great variety of functions.
- I2C 32 Channel PWM / Servo Board: - The interface board which sends turns the data from the raspberry pi into PWM signals for the 24 radio-controlled servos.
- WI-FI Dongle: - This is the unit, which enables the user to control the spider robot wirelessly.
- Webcam: - This is the eye of the spider robot in tis project.
- Servos: - These are the motors controlled by the interface board in order to make the robot walk.
- A computer machine with minimum of 1 gigabyte RAM, Wi-Fi Card, and windows 7 operating system, 2 USB or a Mobile Device with Wi-Fi. This will be used as the client system.

4.2.1 Building the Spider Robot Skeleton

As mentioned in the above section the skeleton of the spider robot was built using the chopsticks spider robot model. A detailed instruction of how to make the spider robot skeleton is in Appendix D. Once the base of the spider robot was made as shown as in figure 3-2, the spider robot was tested and was found that when it lifts one leg it didn't lift completely of the ground since the weight forced the leg keen down.

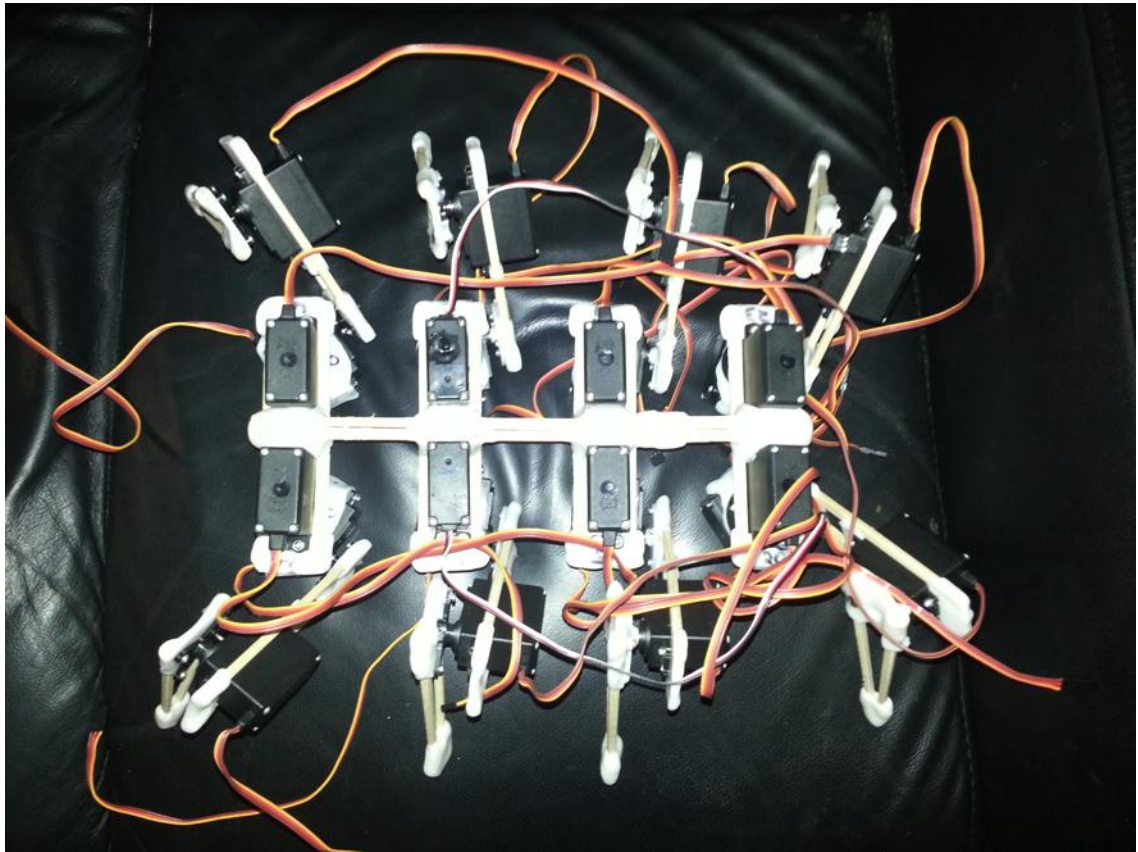


Figure 4-2: Base model of the spider robot

So to solve this problem a chopstick was stuck on both the sides of the spider robot so that the leg will be lifted off the ground. The modified base of the spider robot is shown in figure3-3.

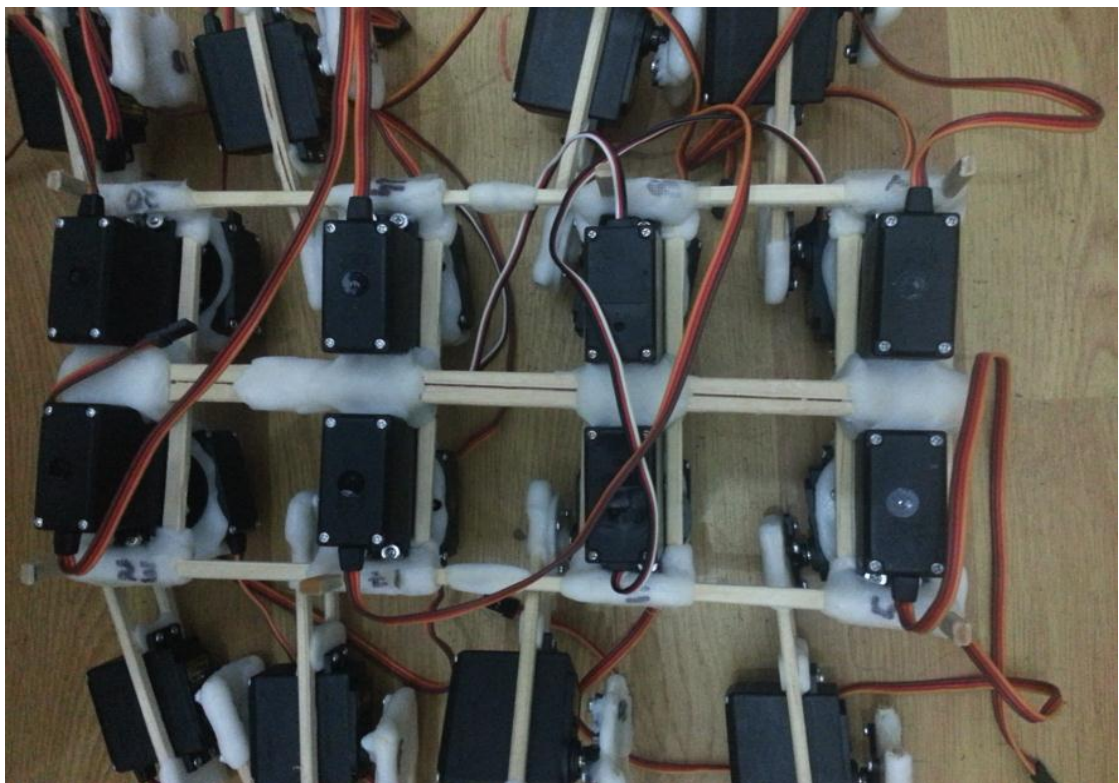


Figure 4-3: Modified base of the spider robot

Once the base of the spider robot was all done. The next thing was to create a top structure onto the base model so that the electronic components could be fixed onto it. The top structure design was a simple rectangular shape frame, with five divisions and a webcam servo-mounting frame in front as shown as in figure 3-4.



Figure 4-4: Top Structure of spider robot

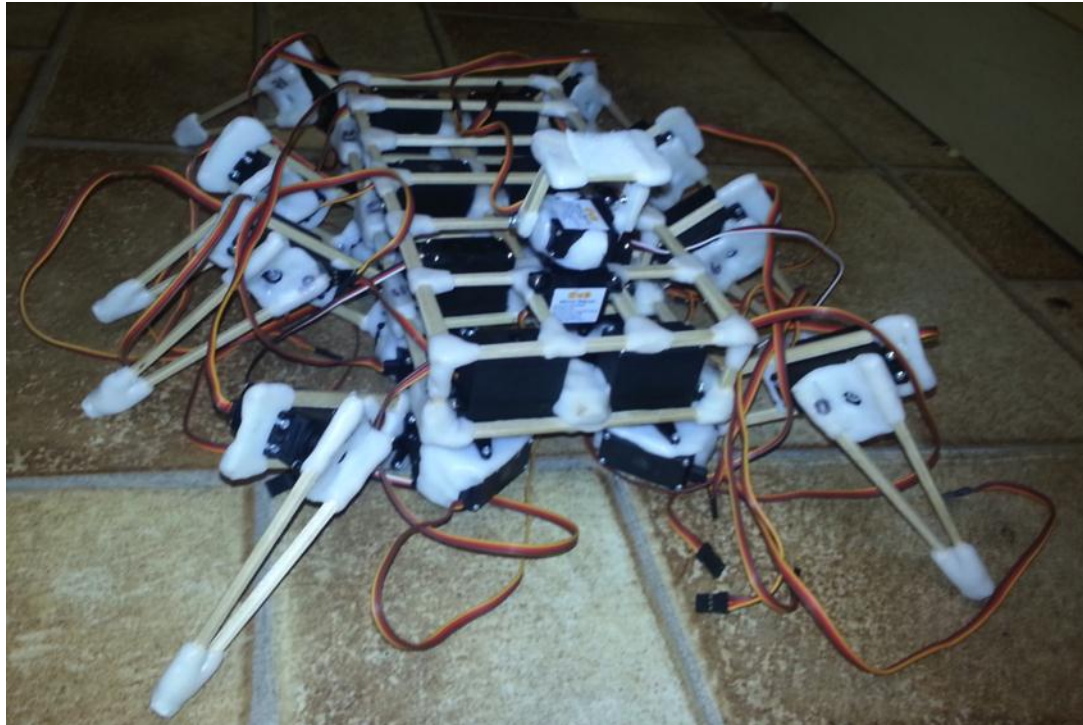


Figure 4-5: Finished model of the spider robot

Figure 3-5 shows the finished model of the spider robot with the top structure attached to the base model and the webcam stand mounted.

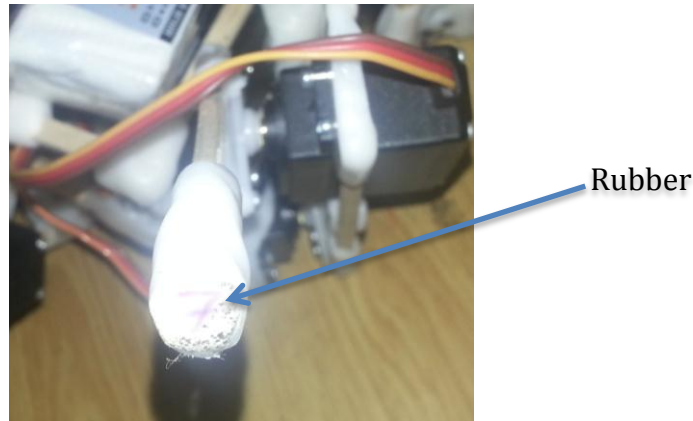


Figure 4-6: Modified Leg

Figure 3-6 shows the modification that was done on each leg to improve the grip of walking. The polymorph is heated at the end of each leg and a small piece of rubber is placed on it. This modification was done due to the lack of grip on some surfaces, which made the spider robot hard to walk. Making this modification has made the spider robot to walk on most of the surfaces.

4.2.2 Connections

Once the spider robot skeleton is fully made the next stage is to make all the connections.

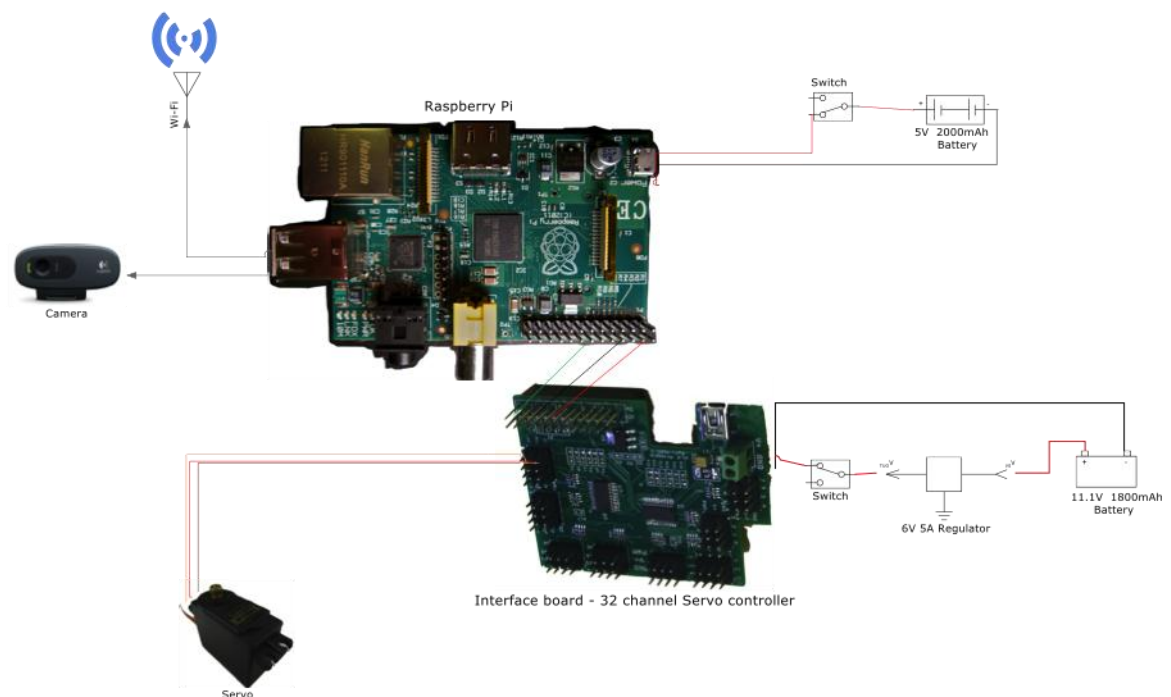


Figure 4-7: Connections

The connections are done as shown in the above figure 3-6. The 24 servos on the finished model are connected to the interface board – I2C 32 channel servo board which is plugged into the GPIO port of the raspberry pi. Wi-Fi dongle and the camera are plugged into the USB port of the pi. The Webcam is mounted onto the webcam stand. Two separate batteries power the raspberry pi and the interface board, since the servos can cause disturbances on the battery line and also the interface board needs more current to power the servos. The raspberry pi is power with a 5V 2000mAh battery. And the servos are powered with a 11.1V 1800mAh battery through a 6V 5A regulator to limit the voltage to 6V.

4.2.3 Testing Servos

Each servo was tested for their performance, speed and functionality with a Turnigy Servo Tester as shown as in figure 4-7.



Figure 4-8: Testing servos with Turnigy Servo Tester

The 24 radio-controlled servos (16 Tower Pro MG995 and 8 Tower Pro SG5010) were tested using the turnigy servo tester.

4.3 Software Implementation

The software tools that are required for the implementation of the proposed project are listed below

- A optimised version of Debian on the raspberry pi to support programming in python for the servos.
- VNC server to access the raspberry pi remotely.
- Python for the programming or the development of the program to control the servo motors.
- HTML and JavaScript for receiving and sending data from the raspberry pi and also for developing the server and client systems.
- Web server to stream the video across to the client system.

Next in this section we are discussing about the setups that are required for the implementation of the project.

4.3.1 Setting up Raspberry Pi

The setup for the raspberry for this project is divided into three: -

- Installing Raspbian “wheezy”
- Installing VNC Server
- Wi-Fi connection

4.3.1.1 Installing Raspbian “wheezy”

The raspbian is a customized version of Debian built for the raspberry pi. It contains LXDE, Midori and other development tools.

1. The image of the raspbian is downloaded from the raspberry pi website under downloads.
2. The image file comes in a zipped file, which is unzipped and extracted, to the SD card using Win32DiskImager software.
3. Once the files are extracted to the SD card it is ready for start-up.
4. The SD card is plugged into the raspberry pi and powered on. After a few seconds the raspbian loads up.

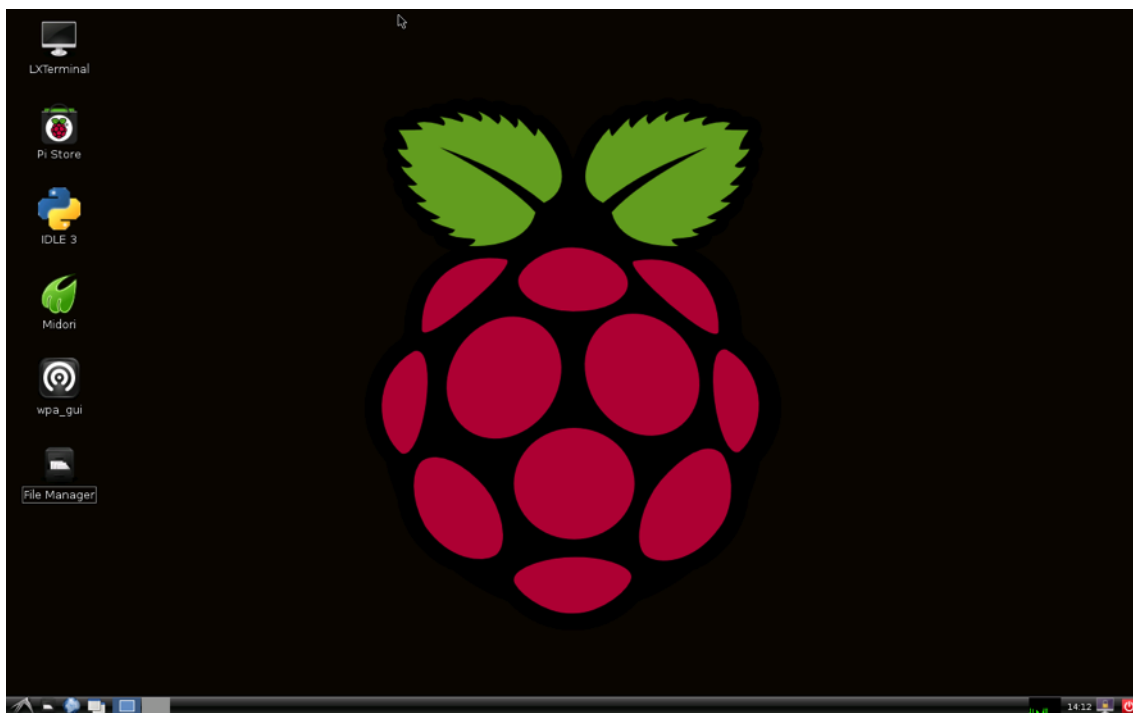


Figure 4-9: Raspbian installed on raspberry pi

4.3.1.2 Installing VNC Server

VNC server is a server hosted on the raspberry pi, which enables another computer to start a session remotely on the raspberry pi.

1. To install VNC server, first log into the pi and install the Tight VNC Package

```
$ sudo apt-get install tightvncserver
```

1. Now run the tightVNC Server and setup a new password

```
$ tightvncserver
```

1. Now that its installed, start a VNC.

```
$ vncserver :0 -geometry 1920x1080 -depth 24
```

This will starts a session on VNC display zero (:0)

1. Once tight VNC server is running, the started session on the raspberry pi can be accessed from a computer using a screen sharing software as shown as in figure 4-9. Entering the IP address of the raspberry pi and the correct port gains access to the session.

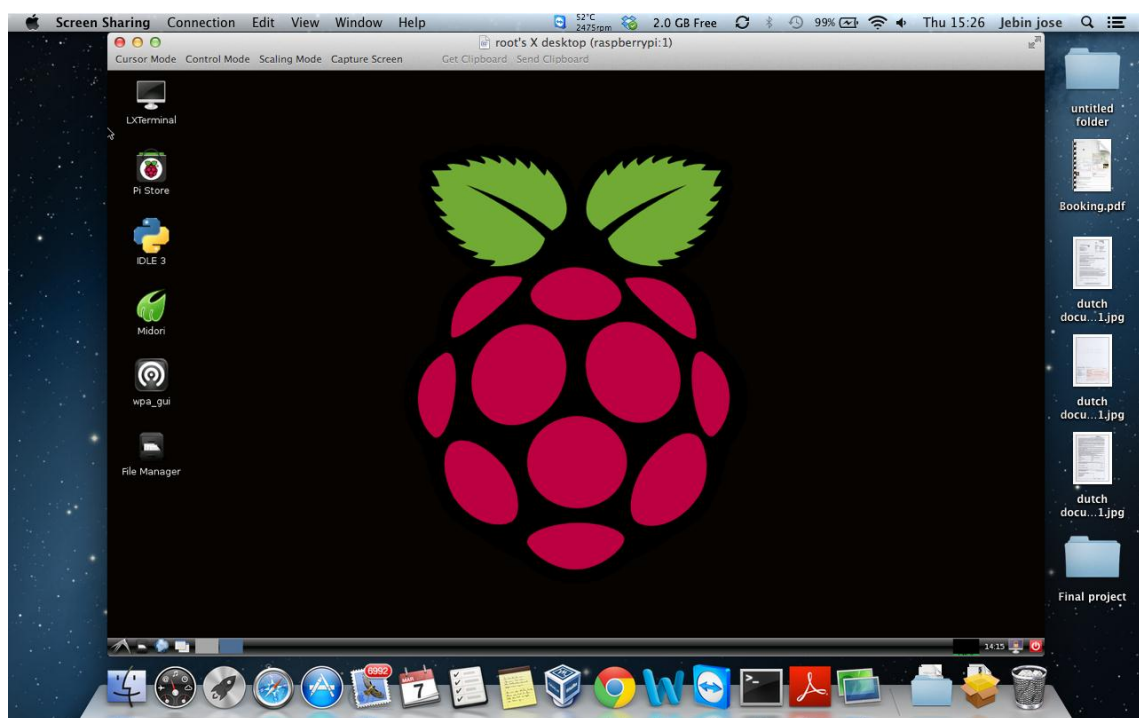


Figure 4-10: VNC Display Session

4.3.1.3 Wi-Fi connection

The Wi-Fi connection was setup with the application wpa_gui on the raspbian. The application allows scanning for Wi-Fi signals and connecting to it and also reconnects to it when rebooted.

4.3.2 Setting up Interface board

Since the raspberry pi has only one PWM output, it was no good for the project on its own. But the raspberry pi had I2C outputs, which can be used to communicate with the PCA9685 used on the I2C 32 channel PWM/Servo Board. For the interface board to work with the raspberry pi, it had to be configured for I2C connection.

2. To enable the I2C for use in raspbian,

1. open LXTerminal and enter:

```
sudo nano /etc/modules
```

2. Then add these two lines to the modules

```
i2c-bcm2708
```

i2c-dev

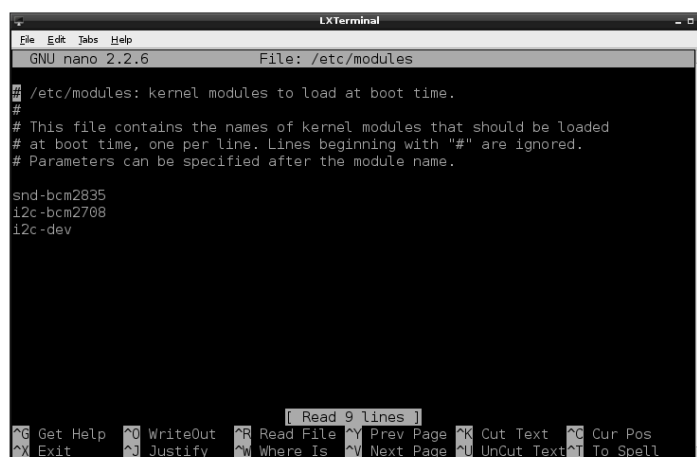


Figure 4-11: Modules

3. Reboot the system after editing the file.

4. Then comment two lines in file `/etc/modprobe.d/raspi-blacklist.conf`

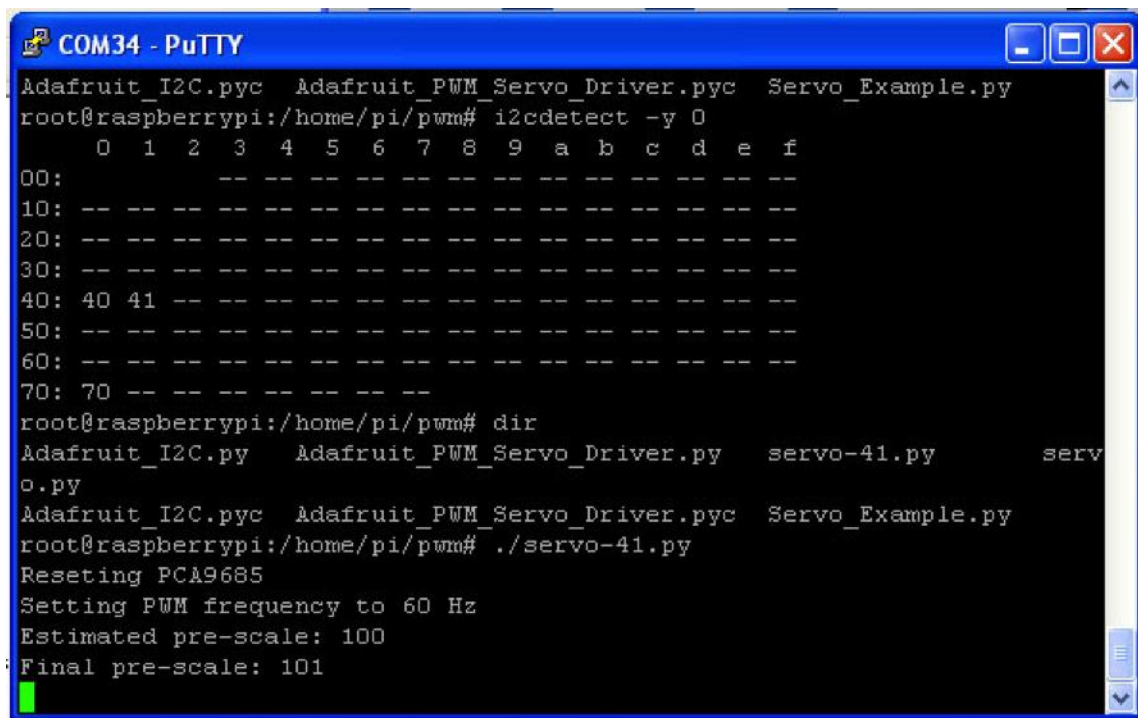
If the file is not present in the system, then there is nothing to do. If the file exists comment the following two lines.

```
blacklist spi-bcm2708
```

```
blacklist i2c-bcm2708
```

5. Once everything is done, check the connected device by typing the following command.

```
sudo i2cdetect -y 1
```



```
COM34 - PuTTY
Adafruit_I2C.pyc  Adafruit_PWM_Servo_Driver.pyc  Servo_Example.py
root@raspberrypi:/home/pi/pwm# i2cdetect -y 0
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40: 40 41 --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70: 70 --  --  --  --  --  --  --  --  --  --  --  --  --  --
root@raspberrypi:/home/pi/pwm# dir
Adafruit_I2C.py  Adafruit_PWM_Servo_Driver.py  servo-41.py  serv
o.py
Adafruit_I2C.pyc  Adafruit_PWM_Servo_Driver.pyc  Servo_Example.py
root@raspberrypi:/home/pi/pwm# ./servo-41.py
Resetting PCA9685
Setting PWM frequency to 60 Hz
Estimated pre-scale: 100
Final pre-scale: 101
```

Figure 4-12: Shows the I2C devices found

This shows the two i2c-detected devices in the system (40,41).

- Now that the I2C is enabled for use in raspbian, download the Code from Github to configure the servo driver.

```
$ git clone https://github.com/adafruit/Adafruit-
Raspberry-Pi-Python-Code.git
$ cd Adafruit-Raspberry-Pi-Python-Code
$ cd Adafruit_PWM_Servo_Driver
```

4.3.3 Setting up Webcam

The webcam used for the project was a Logitech C270, which was compatible with raspberry pi with few setups done on the pi. To setup the webcam for this project the following task are done: -

4.3.3.1 Installing the camera

- First UVC support to the image was added by running the following command: -
`sudo apt-get install rpi-update`

This will update the missing UVC support onto the image. Once this is done reboot the pi.

2. Now update the packages by running the following command: -

```
sudo apt-get update
sudo apt-get upgrade
```

3. Next the guvcview webcam viewer is installed by running the following command: -

```
sudo apt-get install guvcview
```

4. Once the guvcview webcam viewer is installed, the permissions are setup and the driver is enabled by running the following command: -

```
sudo usermod -a -G video pi
sudo modprobe uvcvideo
```

5. Once the driver is enabled reboot the Pi and open the webcam by running the following command: -

```
guvcview
```

This will show the webcam images in a window.

4.3.3.2 Setting up Webcam Server

Once the webcam is installed and working properly the webcam server using a mjpg streaming is setup for the video streaming by following the instructions: -

1. First the following packages are installed by running the following command: -

```
apt-get install libv4l-dev
apt-get install libjpeg8-dev
apt-get install subversion
apt-get install imagemagick
```

2. Now the mjpg streaming is downloaded and compiled by running the following command: -

```
svn co https://mjpg-streamer.svn.sourceforge.net/svnroot/mjpg-streamer mjpg-streamer
cd mjpg-streamer/mjpg-streamer
```

```
make USE_LIBV4L2=true clean all
```

```
make DESTDIR=/usr install
```

3. Now that the mjpg streamer is installed running the following command can start the mjpg stream at port 8090 with a resolution of 320×240 px and 10 frames per second: -

```
mjpg_streamer -i "/usr/lib/input_uvc.so -d /dev/video0  
-r 320x240 -f 10" -o "/usr/lib/output_http.so -p 8090 -w  
/var/www/mjpg_streamer"
```

where -d: device

-r: resolution

-f: frame rate

-p: port

-w: web serving directory

4. To use the web server the mjpg streamer has to be started as shown above and the following added to the html file in order to stream the video onto a web page: -

```
img src="http://ipaddress of pi:8090/?action=stream"  
width="752"
```

4.3.4 Coding the Servos

The coding was completely done in Python. Python is a simple programming language, which was used to program each functions of the spider robot.

4.3.4.1 Frequency/pulse length

The pulse length and the frequency are set using the code shown below: -

```
def setServoPulse(channel, pulse):
```

```
    pulseLength = 1000000    # 1,000,000 us per second  
    pulseLength /= 60        # 60 Hz  
    print ("%d us per period" % pulseLength)  
    pulseLength /= 4096      # 12 bits of resolution  
    print ("%d us per bit" % pulseLength)  
    pulse *= 1000
```

```

pulse /= pulseLength
pwm2.setPWM(channel, 0, pulse)
pwm.setPWM(channel, 0, pulse)

pwm.setPWMFreq(60)          # Set frequency to 60 Hz
pwm2.setPWMFreq(60)

```

4.3.4.2 Initializing PWM devices

The PWM devices, which are the two PCA9685 chips in this project, are initialized using the following code: -

```

pwm = PWM(0x40, debug=True) # Initialize the PWM device using the default
                             address
pwm2 = PWM(0x41, debug=True) # Initialize the pwm2 device using the default
                             address

```

4.3.4.3 Coding the spider robot functions

There are four essential functions for the spider robot, which are forward, backward, right and left. There are other functions too like stand, sit, cam up, cam down, cam right and cam left. But first of all the servos had to be initialized to a position.

4.3.4.3.1 Initializing

To initial the spider robot to a sitting position, each servo were calibrated to the correct position and the pulse was recorded down against the servos pin. The recorded pulse was then entered into a function called initialize as shown in the code below: -

```

def initialize():

    pwm2.setPWM(8, 0, 400)    # camera servo initialize
    pwm2.setPWM(9, 0, 400)    # camera servo initialize
    pwm.setPWM(0, 0, 345)     # servo 0 initialize
    pwm.setPWM(1, 0, 420)     # servo 1 initialize
    .
    .
    .
    .
    pwm2.setPWM(6, 0, 295)    # servo 22 initialize
    pwm.2setPWM(7, 0, 300)    # servo 23 initialize

```


4.3.4.3.2 Functions

Each function was implemented using the designs shown in the above chapter. For example the forward zigzag walking was implemented into a function called forward as shown as in the code below: -

```
def forward(n):      # function forward has 1 parameter n

    j = 0              # j is initialized to zero
    standing()         # function standing is called
    while j < n:      # a while is created so that if j is less the n the loop keeps
                        running

        pwm.setPWM(0, 0, 395)
        pwm.setPWM(1, 0, 440)
        pwm.setPWM(9, 0, 420)
        pwm.setPWM(10, 0, 450)
        pwm.setPWM(12, 0, 310)
        pwm.setPWM(13, 0, 300)
        pwm2.setPWM(5, 0, 485)
        pwm2.setPWM(6, 0, 480)
        time.sleep(w)
        # w is declared globally

        pwm.setPWM(2, 0, 550)
        pwm.setPWM(11, 0, 350)
        pwm2.setPWM(14, 0, 370)
        pwm2.setPWM(7, 0, 375)
        time.sleep(w)

        pwm.setPWM(1, 0, 490)
        pwm.setPWM(9, 0, 470)
        pwm.setPWM(10, 0, 400)
        pwm.setPWM(12, 0, 260)
        pwm.setPWM(13, 0, 350)
        pwm2.setPWM(5, 0, 535)
        pwm2.setPWM(6, 0, 430)
        time.sleep(w)

        pwm2.setPWM(5, 0, 485)
        pwm.setPWM(0, 0, 345)
        pwm.setPWM(2, 0, 500)
        pwm.setPWM(11, 0, 300)
        pwm2.setPWM(14, 0, 320)
        pwm2.setPWM(7, 0, 425)
        #time.sleep(0.5)
```

This part of the code lifts the first set of four legs and keeps it forward and then brings it back to its original servo pulse to give it the forward movement.

```

pwm.setPWM(3, 0, 310)
pwm.setPWM(4, 0, 440)
pwm.setPWM(6, 0, 345)
pwm.setPWM(7, 0, 310)
pwm.setPWM(15, 0, 400)
pwm2.setPWM(0, 0, 490)
pwm2.setPWM(2, 0, 325)
pwm2.setPWM(3, 0, 345)
time.sleep(w)

```

```

pwm.setPWM(5, 0, 200)
pwm.setPWM(8, 0, 240)
pwm2.setPWM(1, 0, 370)
pwm2.setPWM(4, 0, 325)
time.sleep(w)

```

```

pwm.setPWM(4, 0, 390)
pwm.setPWM(6, 0, 295)
pwm.setPWM(7, 0, 360)

```

```

pwm.setPWM(15, 0, 450)
pwm2.setPWM(0, 0, 440)
pwm2.setPWM(2, 0, 275)
pwm2.setPWM(3, 0, 395)
time.sleep(w)

```

```

pwm2.setPWM(2, 0, 325)
pwm.setPWM(3, 0, 360)
pwm.setPWM(5, 0, 250)
pwm.setPWM(8, 0, 290)
pwm2.setPWM(1, 0, 420)
pwm2.setPWM(4, 0, 275)
#time.sleep(0.5)

```

```

j = j + 1      # j is incremented

```

This part of the code lifts the second set of four legs and keeps it forward and then brings it back to its original servo pulse to give it the forward movement.

The forward function works by setting each servos to a pulse needed to perform the forward task and then delaying for a millisecond and running the next set and so on. The designs discussed in the previous chapter for walking methods are implemented as different functions. The full spider robot code is attached in Appendix A.

4.3.4.3.3 Testing

Each function was tested in different condition to see if there are any glitches in the code. Problems were rectified and tested again in different conditions.

4.3.5 Web interface

The web interface design discussed in the pervious chapter was turned into a full color mockup of the interface as shown as in figure 4-13.



Figure 4-13: Initial Web Interface

As seen from the above figure the web interface has a live camera feed and 10 main controls. The web interface is developed in HTML and JavaScript language. The web interface is hosted on the raspberry pi using a Pico server. Pico is a bridge between server side Python and client side JavaScript. Pico is a server, a Python library and a Javascript library. The server is a WSGI application which can be run standalone or behind Apache with mod_wsgi[14]. The web interface works using Pico, when the forward button is pressed the Pico loads the python file Spider and a JavaScript calls the function `Spider.forward()` as shown in the code below.

```
pico.load('Spider');
pico.main = function() {
  document.getElementById('forward').addEventListener('click'
, function(){Spider.forward(4);
});
```

The web interface code is attached in Appendix B.

4.3.6 Server start-up

There are three servers in total to be start-up at boot time. There are the VNC server, Webcam server and the Pico server. All the three server start-ups can be done by creating a shell script in /etc/init.d with an appropriate name such as camboot, vncboot or picoboot. The full instruction on how to create the file and enable dependency based boot sequencing is attached in Appendix C.

4.4 Implemented System

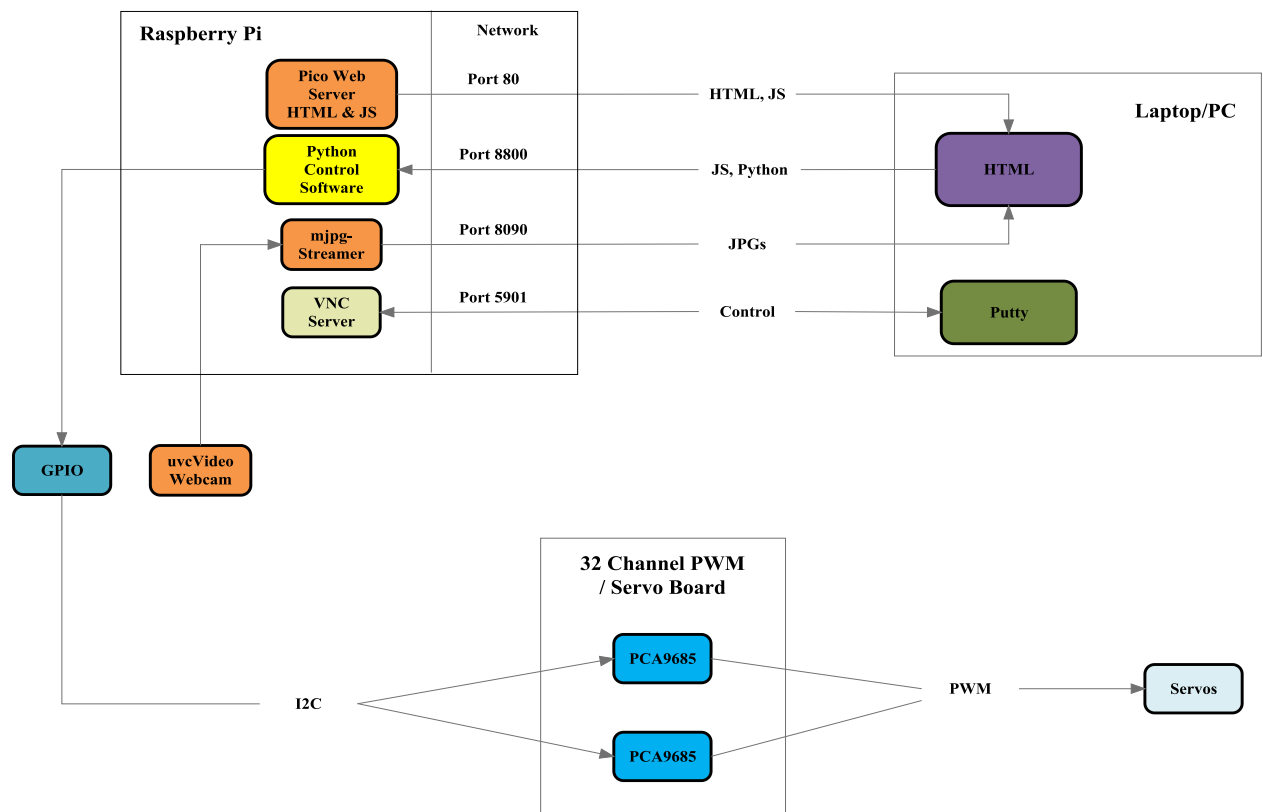


Figure 4-14: Overall system

Figure 4-14 shows the overall structure of the implemented system and how the different devices are connected together.

4.5 Flow chart

The flow chart shows the algorithm of the proposed system. With the help of the Flow chart we can understand the algorithm easily.

4.5.1 Server flowchart

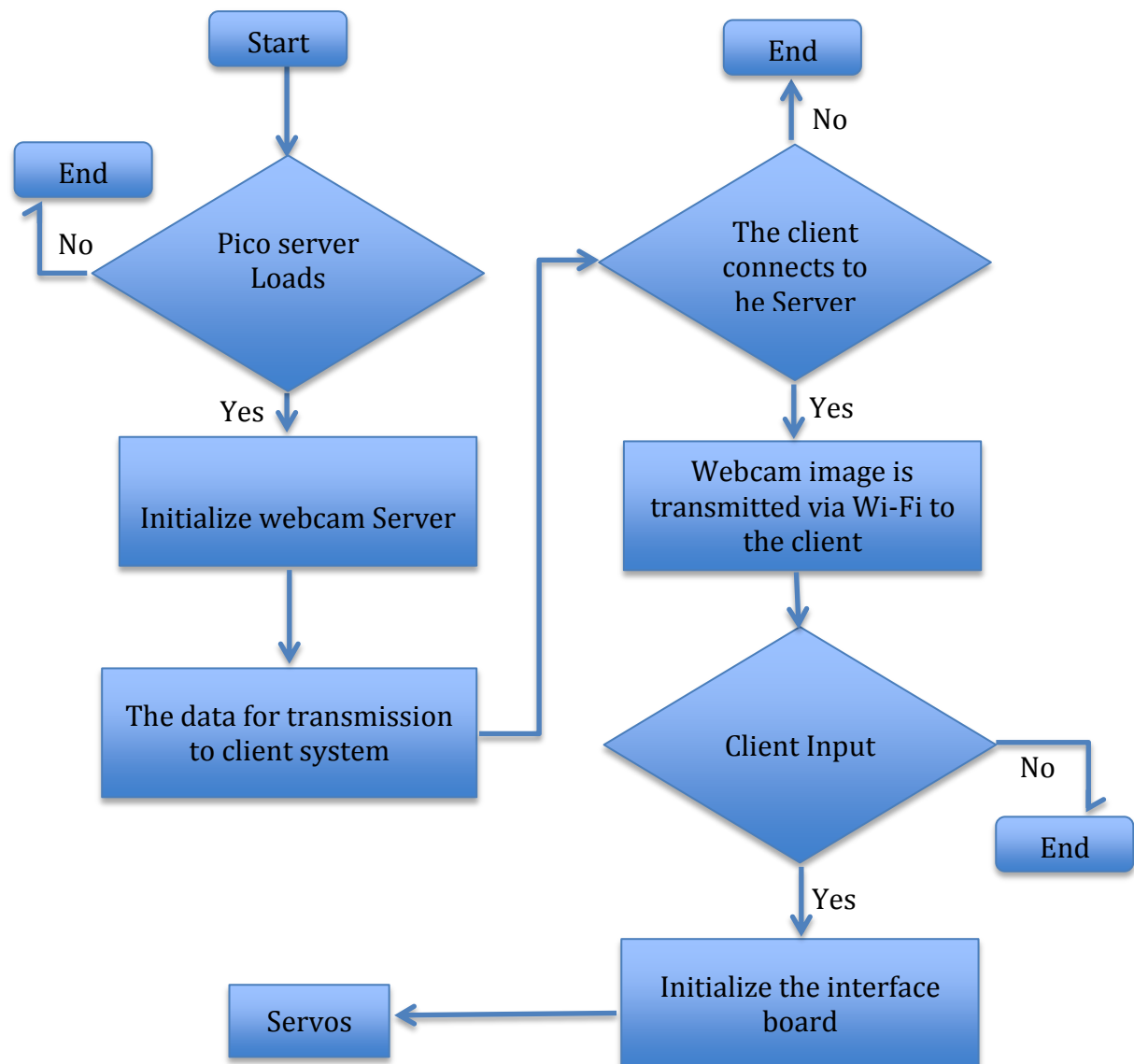


Figure 4-15: Server flow chart

4.5.2 Client Flowchart

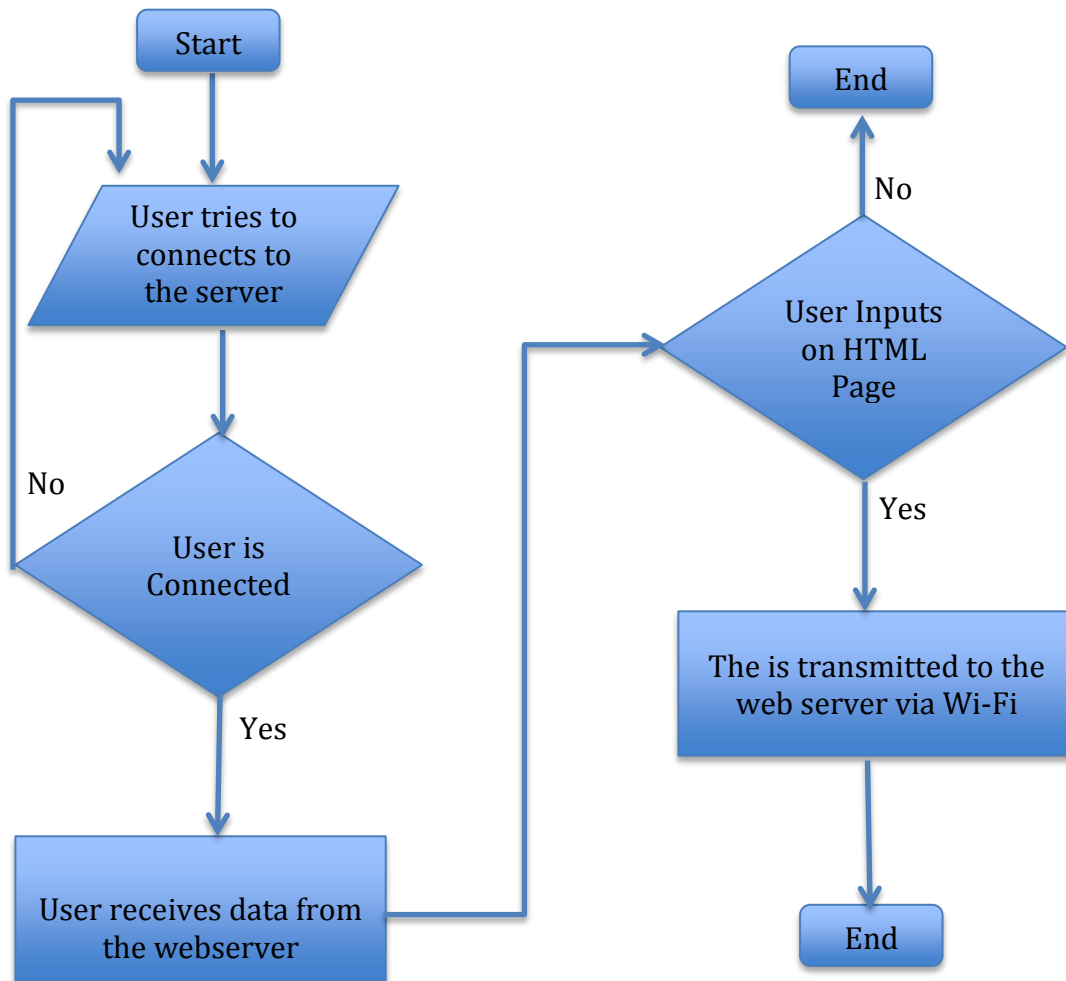


Figure 4-16: Client flow chart

Chapter 5

5. Results

In this section we go through the test results of the test that were done on the Unit, web interface and the result of the spider robot.

5.1 Unit Test

Carrying out unit test is kind of a pre-release of the system. Performing unit test in presence of supervisor ensures that final product had met all requirements. Unit test results listed described in below showing table:

Test case description	Test result	Comment
Webcam image display	Accepted	
Sit	Accepted	
Stand	Accepted	
Move Forward	Accepted	
Move Backward	Accepted	
Turn Right	Accepted	
Turn Left	Accepted	
Webcam Rotation	Accepted	Is a bit unstable.

Table 5-1: Unit test results.

The recommended changes shows on the comment tab is not yet implemented as it require extra time to add these features and these are not affecting the services of the system

5.2 Web Interface test results

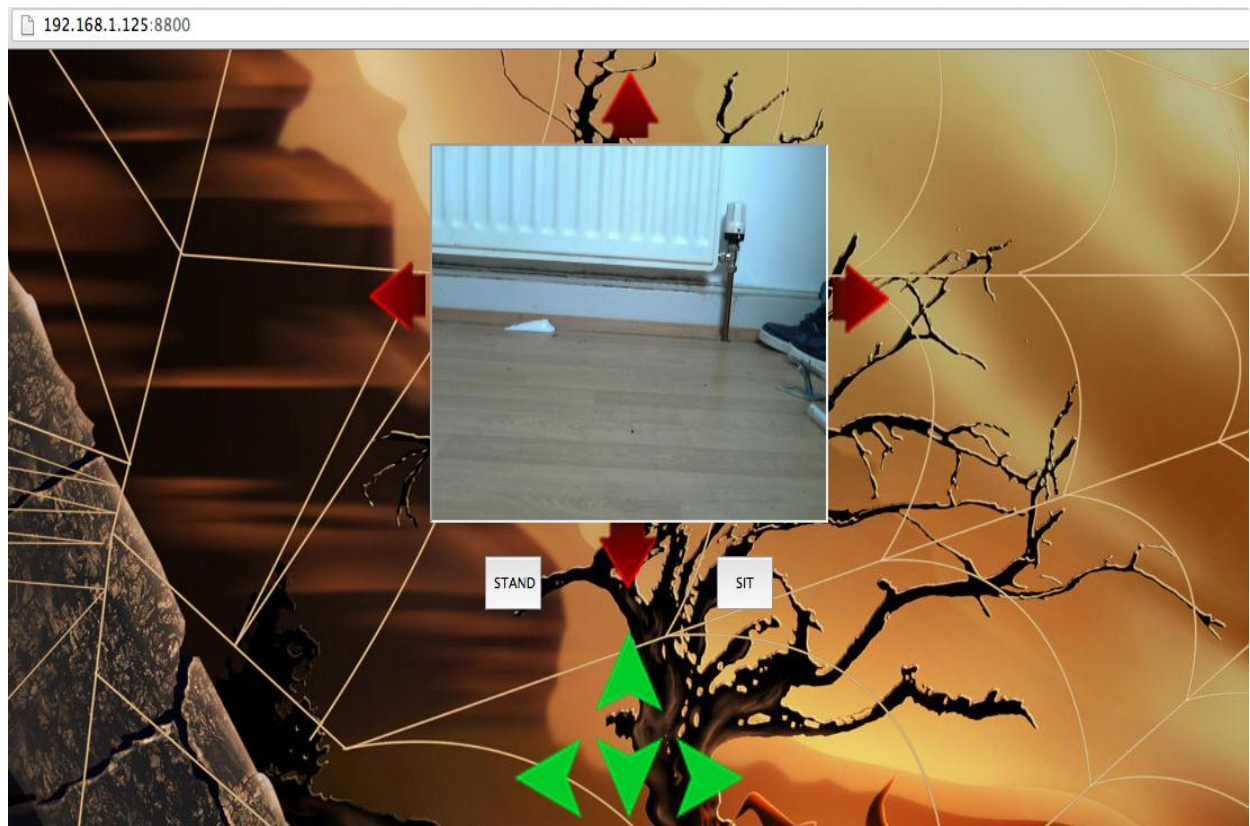


Figure 5-1: Client system test result

As we see from the above Figure 5-1 the client system gets connected to the server system through an IP address (192.168.1.125:8800) and is receiving the camera images but since the data size is too big, it can take a bit longer to transmit it to the client interface. The system works perfectly fine and sends the users instructions back to the spider robot, which is then executed.

5.3 Spider robot

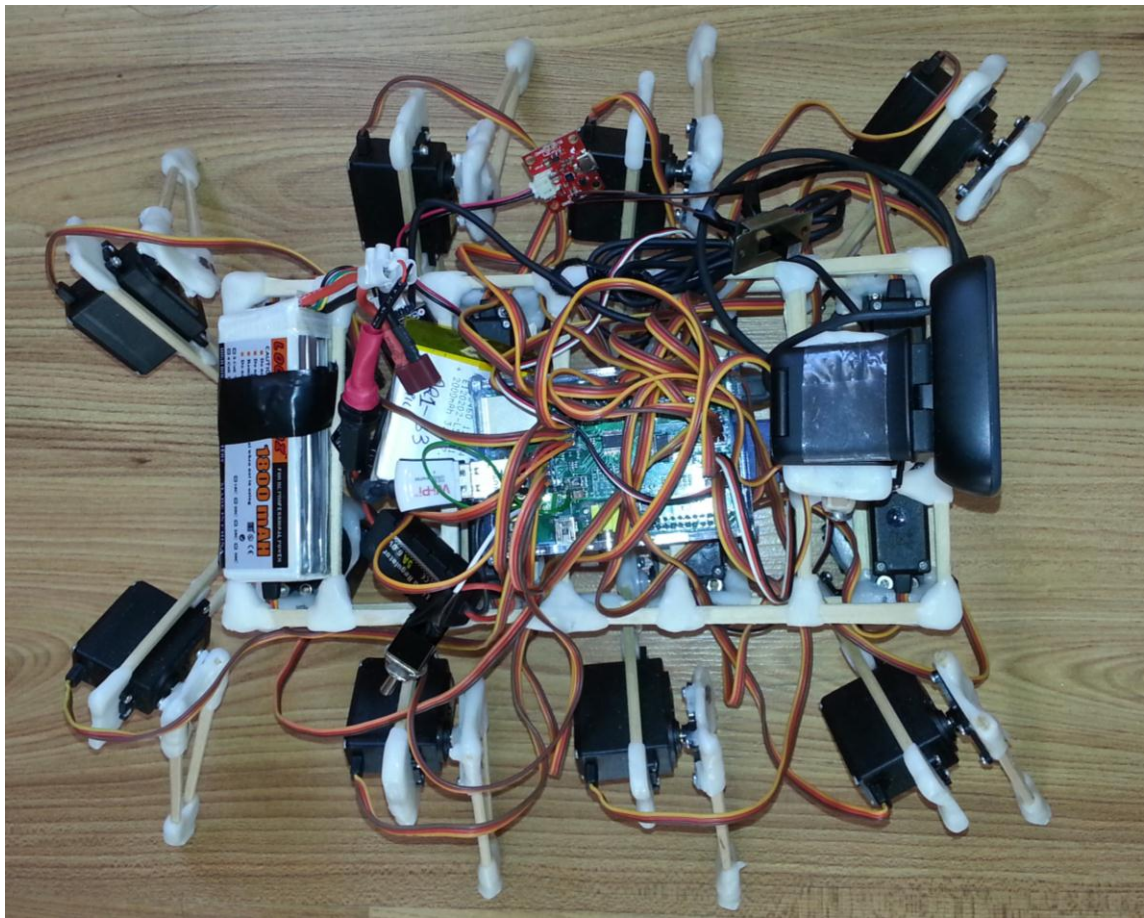


Figure 5-2: Spider robot top view

The spider robot, which is shown in figure 5-2, is in its sit position. The robot has succeeded in meeting its requirements and can walk forward, backwards, turn left, turn right, sit, stand and even rotate its camera around. At the time of the tests the body section was not completely done as seen in the above figure. The raspberry pi and batteries have to be fitted on to the mainframe of the robot and sort the cables with a cable tie.

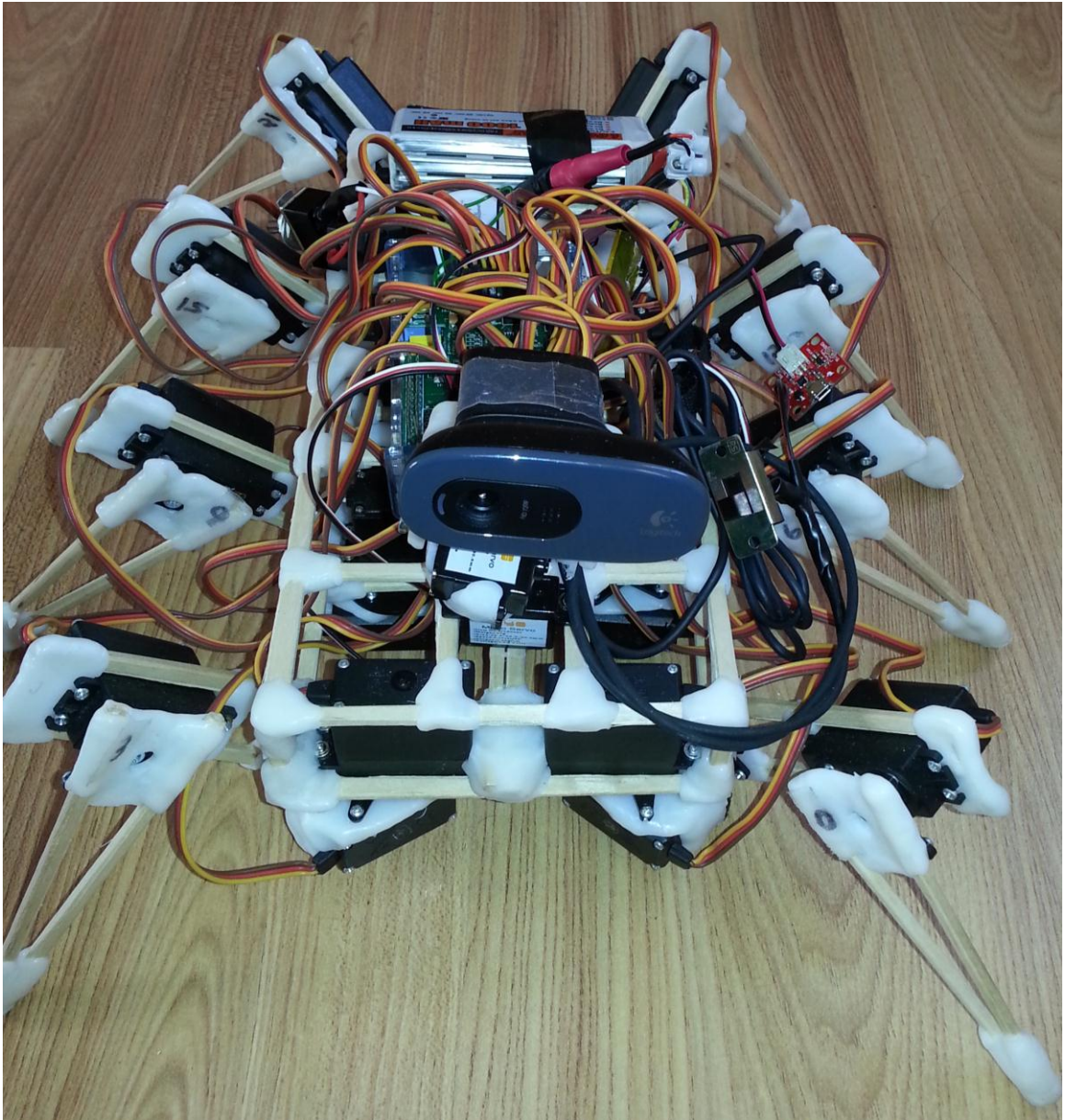


Figure 5-3: Spider robot front view

Chapter 6:

6. Discussion / Conclusions / Future Work

6.1 Discussion

During the whole period of the project I gained a lot of knowledge on the raspberry pi, servos and programming in python. If we talk about the achievements out of the project when starting to do the project it was to control the servos using raspberry pi on a robot and transmit that data via any wireless technology to another device and able to collect the data and control the robot or raspberry pi in a real time instance. Out of which all the work was completed. The main achievements that I gained out of the project were that I got to learn programming in python and could learn how to program a user interface web page. Another main achievement was I could learn and understand the raspberry pi technology and the wide applications of raspberry pi. There are lots of many other areas where the raspberry pi could be used for robotic applications and that are the reasons for me to choose this project.

6.2 Conclusions

To get to the aim of a project there will be always a set of objectives, to achieve that objectives we need to know how where and with what resource is the step towards completing the objectives taken. Now in this project too to get to the aim of the project there was a set of objectives, which gradually changed as the project research was completed and then while testing a certain technology the objectives again changed due to the failure of the method. Now the first thing of the project is a good research, I had to do a wide and a strong research before I started to put my objectives as this technology was new in market.

The research for the project was done using Advanced Google search and also from the search engines available in the student portal like e-search, science direct etc.

The Google advanced search is the one that was more widely used as it is a new technology and there are very less articles or journals published regarding the raspberry pi technology.

Each stage of the project was tested after every part of it was completed and then moved on to the next one. During the course of the project I gained knowledge of python programming which is more simple and easy to use than the c++. I also gained knowledge of the raspberry pi technology and what the small computer is capable of. After knowing the capabilities of raspberry pi and the applications it could have in the field of robotics, it actually has made me to think of doing more research work on the raspberry pi for the robotic applications.

The challenges that I faced during the course of the project were that of the time constrain, as I had to learn about the raspberry pi and servos technology and then learn programming in python and HTML. Then during the programming of the server client interfaces the problems of calling functions with a button press. One of the main challenges that I could not overcome till the time of the reporting was the power failure of a PCA9685 chip when some functions are called. Other than the small problem the spider robot works fine and meets all its purpose.

If given an opportunity to work again on the same technology i.e. the raspberry pi and servo technology or on a project like this where the raspberry pi is used for any kind off application I would be happy to take it up.

6.3 Future Work

In the future this raspberry pi technology can be used in various different fields of work. It also can be setup to go a warzone or a disaster site where the people cannot reach and record images.

The spider robot can be made autonomous with the help of ultrasonic sensor, gyroscope, compass and a GPS. So that it can be set to a target or a specific area where in can monitor. The robot can also be developed into an advanced robot toy for young people. This technology with a infrared camera can also be used as security cameras where there is only little light.

Reference

1. Wikipedia, Raspberry pi, Available at:
http://en.wikipedia.org/wiki/Raspberry_Pi (Accessed: 10/02/2013).
2. Microchip, MPLAB, 1998, Available at:
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002 (Accessed: 29/03/2013).
3. CDC, Polymorph, Available at: www.c-d-c-shop.com/Products/Polymorph/ (Accessed: 15/02/2013).
4. Seattle Robotics Society, Servos, Available at:
<http://www.seattlerobotics.org/guide/servos.html> (Accessed: 10/02/2013).
5. Wikipedia, PWM, Available at: http://en.wikipedia.org/wiki/Pulse-width_modulation (Accessed: 10/02/2013).
6. Wikipedia, python, Available at:
[http://en.wikipedia.org/wiki/Python_\(programming_language\)](http://en.wikipedia.org/wiki/Python_(programming_language)) (Accessed: 15/02/2013).
7. WINAVR, Radio-controlled servos, Available at:
<http://winavr.scienceprog.com/example-avr-projects/servo-motor-control-using-avr.html> (Accessed: 10/02/2013).
8. Jeff Skinner box, GPIO pins, Available at:
<http://jeffskinnerbox.wordpress.com/2012/12/05/raspberry-pi-serial-communication/raspberry-pi-rev-1-gpio-pin-out-2/> (Accessed: 14/2/2013).
9. Wikipedia, Robotics, Available at: <http://en.wikipedia.org/wiki/Robotics> (Accessed: 17/02/2013).
10. Trossen Robotics, Hexapod, 2006, Available at:
<http://www.trossenrobotics.com/phantomx-ax-hexapod.aspx> (Accessed: 17/04/2013).
11. Wikipedia, Lipo battery, Available at:
https://en.wikipedia.org/wiki/Lithium_polymer_battery (Accessed: 21/04/2013).
12. Miller, Bradley N (2007). Computer Science : the Python programming language

13. Pridopia, I2C 32 channel PWM servo board, Available at:
<http://www.pridopia.co.uk/pi-9685-2-lp.html> (Accessed: 28/04/2013).
14. Adafruit, 16 channel Servo driver, Available at:
<http://learn.adafruit.com/adafruit-16-channel-servo-driver-with-raspberry-pi/overview> (Accessed: 16/05/2013).
15. Elinux, VNC Server, Available at http://elinux.org/RPi_VNC_Server (Accessed: 12/03/2013).
16. Github, Pico, Available at: <https://github.com/fergalwalsh/pico> (Accessed: 15/05/2013).
17. Just Robot, Video Streaming, Available at: <http://www.justrobots.net/?p=97> (Accessed: 18/05/2013).
18. Github, Pico, Available at: <https://github.com/mirceageorgescu/raspi-tank> (Accessed: 23/03/2013).
19. Make: Project, Chopsticks the spider robot, Available at:
<http://blog.makezine.com/projects/chopsticks-the-spider-robot/> (Accessed: 16/05/2013).
20. NXP, PCA9685, Available at:
http://www.nxp.com/documents/data_sheet/PCA9685.pdf (Accessed: 8/04/2013)

Bibliography

1. Robe Reese, Robert B. (Robert Bryan). (2009). Microcontrollers : from assembly language to C using the PIC24 family
2. Codecademy, Python, Available at:
<http://www.codecademy.com/courses/introduction-to-python-6WeG3/1/1#!/exercises> (Accessed: 27/04/2013).
3. Younkin, George W. (1996). Industrial servo control systems : fundamentals and applications

Appendix A

Zigzag Walking Method

This is the code for the zigzag walking method: -

```
#!/usr/bin/python

from Adafruit_PWM_Servo_Driver import PWM

import time
import pico

#
=====
==
# Example Code
#
=====
==

# Initialise the PWM device using the default address
# bmp = PWM(0x40, debug=True)
pwm = PWM(0x40, debug=True)
pwm2 = PWM(0x41, debug=True)

# n = 3
w = 0.08
servoMin = 150 # Min pulse length out of 4096
servoMax = 600 # Max pulse length out of 4096

def setServoPulse(channel, pulse):

    pulseLength = 1000000          # 1,000,000 us per second
    pulseLength /= 60              # 60 Hz
    print ("%d us per period" % pulseLength)
    pulseLength /= 4096            # 12 bits of resolution
    print ("%d us per bit" % pulseLength)
    pulse *= 1000
    pulse /= pulseLength
    pwm2.setPWM(channel, 0, pulse)
    pwm.setPWM(channel, 0, pulse)

pwm.setPWMPFreq(60)               # Set frequency to 60 Hz
pwm2.setPWMPFreq(60)

def initialize():
```



```
pwm2.setPWM(8, 0, 400) # camera servo initialize
pwm2.setPWM(9, 0, 400) # camera servo initialize
pwm.setPWM(0, 0, 345)
pwm.setPWM(1, 0, 420)
pwm.setPWM(2, 0, 500)
pwm.setPWM(3, 0, 360)
pwm.setPWM(4, 0, 460)
pwm.setPWM(5, 0, 250)
pwm.setPWM(6, 0, 295)
pwm.setPWM(7, 0, 300)
pwm.setPWM(8, 0, 290)
pwm.setPWM(9, 0, 470)
pwm.setPWM(10, 0, 460)
pwm.setPWM(11, 0, 300)
pwm.setPWM(12, 0, 260)
pwm.setPWM(13, 0, 290)
pwm2.setPWM(14, 0, 320)
pwm.setPWM(15, 0, 450)
pwm2.setPWM(0, 0, 500)
```

```
pwm2.setPWM(1, 0, 420)
pwm2.setPWM(2, 0, 275)
pwm2.setPWM(3, 0, 315)
pwm2.setPWM(4, 0, 275)
pwm2.setPWM(5, 0, 535)
pwm2.setPWM(6, 0, 510)
pwm2.setPWM(7, 0, 425)
time.sleep(1)
```

```
pwm.setPWM(1, 0, 380)
pwm.setPWM(4, 0, 500)
pwm.setPWM(7, 0, 260)
pwm.setPWM(10, 0, 500)
pwm.setPWM(13, 0, 250)
pwm2.setPWM(0, 0, 540)
pwm2.setPWM(3, 0, 275)
pwm2.setPWM(6, 0, 540)
```

def standing():

```
pwm.setPWM(0, 0, 345)
pwm.setPWM(1, 0, 490)
pwm.setPWM(2, 0, 500)
pwm.setPWM(3, 0, 360)
pwm.setPWM(4, 0, 390)
pwm.setPWM(5, 0, 250)
pwm.setPWM(6, 0, 295)
pwm.setPWM(7, 0, 360)
pwm.setPWM(8, 0, 290)
pwm.setPWM(9, 0, 470)
pwm.setPWM(10, 0, 400)
pwm.setPWM(11, 0, 300)
```

```
pwm.setPWM(12, 0, 260)
pwm.setPWM(13, 0, 350)
pwm2.setPWM(14, 0, 320)
pwm.setPWM(15, 0, 450)
pwm2.setPWM(0, 0, 440)
pwm2.setPWM(1, 0, 420)
pwm2.setPWM(2, 0, 275)
pwm2.setPWM(3, 0, 395)
pwm2.setPWM(4, 0, 275)
pwm2.setPWM(5, 0, 535)
pwm2.setPWM(6, 0, 430)
pwm2.setPWM(7, 0, 425)
```

```
def camera(n):
```

```
    if (n == 1): # camera up
        pwm2.setPWM(8, 0, 450)
```

```
    elif (n == 2): # camera down
        pwm2.setPWM(8, 0, 350)
```

```
    elif (n == 3): # camera left
        pwm2.setPWM(9, 0, 350)
```

```
    elif (n == 4): # camera right
        pwm2.setPWM(9, 0, 450)
```

```
def forward(n):
```

```
    j = 0
    standing()
```

```
    while j < n:
```

```
        pwm.setPWM(0, 0, 395)
        pwm.setPWM(1, 0, 440)
        pwm.setPWM(9, 0, 420)
        pwm.setPWM(10, 0, 450)
        pwm.setPWM(12, 0, 310)
        pwm.setPWM(13, 0, 300)
        pwm2.setPWM(5, 0, 485)
        pwm2.setPWM(6, 0, 480)
        time.sleep(w)
```

```
        pwm.setPWM(2, 0, 550)
        pwm.setPWM(11, 0, 350)
        pwm2.setPWM(14, 0, 370)
        pwm2.setPWM(7, 0, 375)
        time.sleep(w)
```

```
pwm.setPWM(1, 0, 490)
pwm.setPWM(9, 0, 470)
pwm.setPWM(10, 0, 400)
pwm.setPWM(12, 0, 260)
pwm.setPWM(13, 0, 350)
pwm2.setPWM(5, 0, 535)
pwm2.setPWM(6, 0, 430)
time.sleep(w)
```

```
pwm2.setPWM(5, 0, 485)
pwm.setPWM(0, 0, 345)
pwm.setPWM(2, 0, 500)
pwm.setPWM(11, 0, 300)
pwm2.setPWM(14, 0, 320)
pwm2.setPWM(7, 0, 425)
#time.sleep(0.5)
```

```
pwm.setPWM(3, 0, 310)
pwm.setPWM(4, 0, 440)
pwm.setPWM(6, 0, 345)
pwm.setPWM(7, 0, 310)
pwm.setPWM(15, 0, 400)
pwm2.setPWM(0, 0, 490)
pwm2.setPWM(2, 0, 325)
pwm2.setPWM(3, 0, 345)
time.sleep(w)
```

```
pwm.setPWM(5, 0, 200)
pwm.setPWM(8, 0, 240)
pwm2.setPWM(1, 0, 370)
pwm2.setPWM(4, 0, 325)
time.sleep(w)
```

```
pwm.setPWM(4, 0, 390)
pwm.setPWM(6, 0, 295)
pwm.setPWM(7, 0, 360)
```

```
pwm.setPWM(15, 0, 450)
pwm2.setPWM(0, 0, 440)
pwm2.setPWM(2, 0, 275)
pwm2.setPWM(3, 0, 395)
time.sleep(w)
```

```
pwm2.setPWM(2, 0, 325)
pwm.setPWM(3, 0, 360)
pwm.setPWM(5, 0, 250)
pwm.setPWM(8, 0, 290)
pwm2.setPWM(1, 0, 420)
pwm2.setPWM(4, 0, 275)
#time.sleep(0.5)
j = j + 1
```

```

def backward(n):

    j = 0
    standing()

    while j < n:

        pwm.setPWM(0, 0, 395)
        pwm.setPWM(1, 0, 440)
        pwm.setPWM(9, 0, 420)
        pwm.setPWM(10, 0, 450)
        pwm.setPWM(12, 0, 310)
        pwm.setPWM(13, 0, 300)
        pwm2.setPWM(5, 0, 485)
        pwm2.setPWM(6, 0, 480)
        time.sleep(w)

        pwm.setPWM(2, 0, 450)
        pwm.setPWM(11, 0, 250)
        pwm2.setPWM(14, 0, 270)
        pwm2.setPWM(7, 0, 475)
        time.sleep(w)

        pwm.setPWM(1, 0, 490)
        pwm.setPWM(9, 0, 470)
        pwm.setPWM(10, 0, 400)
        pwm.setPWM(12, 0, 260)
        pwm.setPWM(13, 0, 350)
        pwm.setPWM(0, 0, 345)
        pwm2.setPWM(5, 0, 535)
        pwm2.setPWM(6, 0, 430)
        time.sleep(w)

        pwm.setPWM(2, 0, 500)
        pwm.setPWM(11, 0, 300)
        pwm2.setPWM(14, 0, 320)
        pwm2.setPWM(7, 0, 425)
        #time.sleep(0.5)

    #####

        pwm.setPWM(3, 0, 310)
        pwm.setPWM(4, 0, 440)
        pwm.setPWM(6, 0, 345)
        pwm.setPWM(7, 0, 310)
        pwm.setPWM(15, 0, 400)
        pwm2.setPWM(0, 0, 490)
        pwm2.setPWM(2, 0, 325)
        pwm2.setPWM(3, 0, 345)
        time.sleep(w)

        pwm.setPWM(5, 0, 300)
        pwm.setPWM(8, 0, 340)

```

```

pwm2.setPWM(1, 0, 470)
pwm2.setPWM(4, 0, 225)
time.sleep(w)

```

```

pwm.setPWM(3, 0, 360)
pwm.setPWM(4, 0, 390)
pwm.setPWM(6, 0, 295)
pwm.setPWM(7, 0, 360)
pwm.setPWM(15, 0, 450)
pwm2.setPWM(0, 0, 440)
pwm2.setPWM(2, 0, 275)
pwm2.setPWM(3, 0, 395)
time.sleep(w)

```

```

pwm.setPWM(5, 0, 250)
pwm.setPWM(8, 0, 290)
pwm2.setPWM(1, 0, 420)
pwm2.setPWM(4, 0, 275)
#time.sleep(0.5)
j = j + 1

```

```
def left(n):
```

```

j = 0
standing()

```

```
while j < n:
```

```

pwm.setPWM(0, 0, 395)
pwm.setPWM(1, 0, 440)
pwm.setPWM(9, 0, 420)
pwm.setPWM(10, 0, 450)
pwm.setPWM(12, 0, 310)
pwm.setPWM(13, 0, 300)
pwm2.setPWM(5, 0, 485)
pwm2.setPWM(6, 0, 480)
time.sleep(w)

```

```

pwm.setPWM(2, 0, 450)
pwm.setPWM(11, 0, 350)
pwm2.setPWM(14, 0, 270)
pwm2.setPWM(7, 0, 375)
time.sleep(w)

```

```

pwm.setPWM(0, 0, 345)
pwm.setPWM(1, 0, 490)
pwm.setPWM(9, 0, 470)
pwm.setPWM(10, 0, 400)
pwm.setPWM(12, 0, 260)
pwm.setPWM(13, 0, 350)
pwm2.setPWM(5, 0, 535)
pwm2.setPWM(6, 0, 430)
time.sleep(w)

```

```

    pwm.setPWM(2, 0, 500)
    pwm.setPWM(11, 0, 300)
    pwm2.setPWM(14, 0, 320)
    pwm2.setPWM(7, 0, 425)
    #time.sleep(0.5)

#####

    pwm.setPWM(3, 0, 310)
    pwm.setPWM(4, 0, 440)
    pwm.setPWM(6, 0, 345)
    pwm.setPWM(7, 0, 310)
    pwm.setPWM(15, 0, 400)
    pwm2.setPWM(0, 0, 490)
    pwm2.setPWM(2, 0, 325)
    pwm2.setPWM(3, 0, 345)
    time.sleep(w)

    pwm.setPWM(5, 0, 200)
    pwm.setPWM(8, 0, 340)
    pwm2.setPWM(1, 0, 370)
    pwm2.setPWM(4, 0, 225)
    time.sleep(w)

    pwm.setPWM(3, 0, 360)
    pwm.setPWM(4, 0, 390)
    pwm.setPWM(6, 0, 295)
    pwm.setPWM(7, 0, 360)
    pwm.setPWM(15, 0, 450)
    pwm2.setPWM(0, 0, 440)
    pwm2.setPWM(2, 0, 275)
    pwm2.setPWM(3, 0, 395)
    time.sleep(w)

    pwm.setPWM(5, 0, 250)
    pwm.setPWM(8, 0, 290)
    pwm2.setPWM(1, 0, 420)
    pwm2.setPWM(4, 0, 275)
    #time.sleep(0.5)
    j = j + 1

def right(n):

    j = 0
    standing()

    while j < n:

        pwm.setPWM(0, 0, 395)
        pwm.setPWM(1, 0, 440)
        pwm.setPWM(9, 0, 420)

```

```
pwm.setPWM(10, 0, 450)
pwm.setPWM(12, 0, 310)
pwm.setPWM(13, 0, 300)
pwm2.setPWM(5, 0, 485)
pwm2.setPWM(6, 0, 480)
time.sleep(w)
```

```
pwm.setPWM(2, 0, 550)
pwm.setPWM(11, 0, 250)
pwm2.setPWM(14, 0, 370)
pwm2.setPWM(7, 0, 475)
time.sleep(w)
```

```
pwm.setPWM(0, 0, 345)
pwm.setPWM(1, 0, 490)
pwm.setPWM(9, 0, 470)
pwm.setPWM(10, 0, 400)
pwm.setPWM(12, 0, 260)
pwm.setPWM(13, 0, 350)
pwm2.setPWM(5, 0, 535)
pwm2.setPWM(6, 0, 430)
time.sleep(w)
```

```
pwm.setPWM(2, 0, 500)
pwm.setPWM(11, 0, 300)
pwm2.setPWM(14, 0, 320)
pwm2.setPWM(7, 0, 425)
#time.sleep(0.5)
```

```
#####
```

```
pwm.setPWM(3, 0, 310)
pwm.setPWM(4, 0, 440)
pwm.setPWM(6, 0, 345)
pwm.setPWM(7, 0, 310)
pwm.setPWM(15, 0, 400)
pwm2.setPWM(0, 0, 490)
pwm2.setPWM(2, 0, 325)
pwm2.setPWM(3, 0, 345)
time.sleep(w)
```

```
pwm.setPWM(5, 0, 300)
pwm.setPWM(8, 0, 240)
pwm2.setPWM(1, 0, 470)
pwm2.setPWM(4, 0, 325)
time.sleep(w)
```

```
pwm.setPWM(3, 0, 360)
pwm.setPWM(4, 0, 390)
pwm.setPWM(6, 0, 295)
pwm.setPWM(7, 0, 360)
pwm.setPWM(15, 0, 450)
pwm2.setPWM(0, 0, 440)
pwm2.setPWM(2, 0, 275)
```

```

pwm2.setPWM(3, 0, 395)
time.sleep(w)

pwm.setPWM(5, 0, 250)
pwm.setPWM(8, 0, 290)
pwm2.setPWM(1, 0, 420)
pwm2.setPWM(4, 0, 275)
#time.sleep(0.5)
j = j + 1

```

Alternative Walking Method

This is the code for the alternative walking method: -

```

#!/usr/bin/python

from Adafruit_PWM_Servo_Driver import PWM

import time
import pico
#
=====
==
# Example Code
#
=====
==

# Initialise the PWM device using the default address
# bmp = PWM(0x40, debug=True)
pwm = PWM(0x40, debug=True)
pwm2 = PWM(0x41, debug=True)

w = 0.1
def setServoPulse(channel, pulse):
    pulseLength = 1000000          # 1,000,000 us per second
    pulseLength /= 60              # 60 Hz
    print ("%d us per period" % pulseLength)
    pulseLength /= 4096            # 12 bits of resolution
    print ("%d us per bit" % pulseLength)
    pulse *= 1000
    pulse /= pulseLength
    pwm2.setPWM(channel, 0, pulse)
    pwm.setPWM(channel, 0, pulse)

pwm.setPWMFreq(60)                # Set frequency to 60 Hz

```



```
pwm2.setPWMFreq(60)
```

```
def initialize():
```

```
    pwm.setPWM(0, 0, 345)  
    pwm.setPWM(1, 0, 420)  
    pwm.setPWM(2, 0, 500)  
    pwm.setPWM(3, 0, 360)  
    pwm.setPWM(4, 0, 460)  
    pwm.setPWM(5, 0, 250)  
    pwm.setPWM(6, 0, 295)  
    pwm.setPWM(7, 0, 300)  
    pwm.setPWM(8, 0, 290)  
    pwm.setPWM(9, 0, 470)  
    pwm.setPWM(10, 0, 460)  
    pwm.setPWM(11, 0, 300)  
    pwm.setPWM(12, 0, 260)  
    pwm.setPWM(13, 0, 290)  
    pwm2.setPWM(14, 0, 320)  
    pwm.setPWM(15, 0, 450)  
    pwm2.setPWM(0, 0, 500)  
    pwm2.setPWM(1, 0, 420)  
    pwm2.setPWM(2, 0, 275)  
    pwm2.setPWM(3, 0, 315)  
    pwm2.setPWM(4, 0, 275)  
    pwm2.setPWM(5, 0, 535)  
    pwm2.setPWM(6, 0, 510)  
    pwm2.setPWM(7, 0, 425)  
    time.sleep(1)  
    pwm.setPWM(1, 0, 380)  
    pwm.setPWM(4, 0, 500)  
    pwm.setPWM(7, 0, 260)  
    pwm.setPWM(10, 0, 500)  
    pwm.setPWM(13, 0, 250)  
    pwm2.setPWM(0, 0, 540)  
    pwm2.setPWM(3, 0, 275)  
    pwm2.setPWM(6, 0, 550)
```

```
def standing():
```

```
    pwm.setPWM(0, 0, 345)  
    pwm.setPWM(1, 0, 490)  
    pwm.setPWM(2, 0, 500)  
    pwm.setPWM(3, 0, 360)  
    pwm.setPWM(4, 0, 390)  
    pwm.setPWM(5, 0, 250)  
    pwm.setPWM(6, 0, 295)  
    pwm.setPWM(7, 0, 360)  
    pwm.setPWM(8, 0, 290)  
    pwm.setPWM(9, 0, 470)  
    pwm.setPWM(10, 0, 400)  
    pwm.setPWM(11, 0, 300)  
    pwm.setPWM(12, 0, 260)  
    pwm.setPWM(13, 0, 350)
```

```

pwm2.setPWM(14, 0, 320)
pwm.setPWM(15, 0, 450)
pwm2.setPWM(0, 0, 440)
pwm2.setPWM(1, 0, 420)
pwm2.setPWM(2, 0, 275)
pwm2.setPWM(3, 0, 395)
pwm2.setPWM(4, 0, 275)
pwm2.setPWM(5, 0, 535)
pwm2.setPWM(6, 0, 430)
pwm2.setPWM(7, 0, 425)
time.sleep(1)

```

```
def forward2(n):
```

```
    j = 0
```

```
    while j < n:
```

```

        pwm.setPWM(0, 0, 395)
        pwm.setPWM(1, 0, 440)
        pwm.setPWM(3, 0, 310)
        pwm.setPWM(4, 0, 440)
        pwm.setPWM(12, 0, 310)
        pwm.setPWM(13, 0, 300)
        pwm.setPWM(15, 0, 400)
        pwm2.setPWM(0, 0, 490)
        time.sleep(w)

```

```

        pwm.setPWM(2, 0, 550)
        pwm.setPWM(5, 0, 200)
        pwm2.setPWM(14, 0, 370)
        pwm2.setPWM(1, 0, 370)
        time.sleep(w)

```

```

        pwm.setPWM(1, 0, 490)
        pwm.setPWM(4, 0, 390)
        pwm.setPWM(12, 0, 260)
        pwm.setPWM(13, 0, 350)
        pwm.setPWM(15, 0, 450)
        pwm2.setPWM(0, 0, 440)
        time.sleep(w)

```

```

        pwm.setPWM(0, 0, 345)
        pwm.setPWM(2, 0, 500)
        pwm.setPWM(3, 0, 360)
        pwm.setPWM(5, 0, 250)
        pwm2.setPWM(14, 0, 320)
        pwm2.setPWM(1, 0, 420)
        #time.sleep(0.5)

```

```
#####
```

```
        pwm.setPWM(6, 0, 345)
```

```

pwm.setPWM(7, 0, 310)
pwm.setPWM(9, 0, 420)
pwm.setPWM(10, 0, 450)
pwm2.setPWM(2, 0, 325)
pwm2.setPWM(3, 0, 345)
pwm2.setPWM(5, 0, 485)
pwm2.setPWM(6, 0, 480)
time.sleep(w)

```

```

pwm.setPWM(8, 0, 240)
pwm.setPWM(11, 0, 350)
pwm2.setPWM(4, 0, 325)
pwm2.setPWM(7, 0, 375)
time.sleep(w)

```

```

pwm.setPWM(6, 0, 295)
pwm.setPWM(7, 0, 360)
pwm.setPWM(9, 0, 470)
pwm.setPWM(10, 0, 400)
pwm2.setPWM(2, 0, 275)
pwm2.setPWM(3, 0, 395)
pwm2.setPWM(5, 0, 535)
pwm2.setPWM(6, 0, 430)
time.sleep(w)

```

```

pwm2.setPWM(2, 0, 325)
pwm2.setPWM(5, 0, 485)
pwm.setPWM(8, 0, 290)
pwm.setPWM(11, 0, 300)
pwm2.setPWM(4, 0, 275)
pwm2.setPWM(7, 0, 425)
#time.sleep(0.5)
j = j + 1

```

```

def backward2(n):
    j = 0
    while j < n:
        # Change speed of continuous servo on channel 0

```

```

pwm.setPWM(0, 0, 395)
pwm.setPWM(1, 0, 440)
pwm.setPWM(3, 0, 310)
pwm.setPWM(4, 0, 440)
pwm.setPWM(12, 0, 310)
pwm.setPWM(13, 0, 300)
pwm.setPWM(15, 0, 400)
pwm2.setPWM(0, 0, 490)
time.sleep(w)

```

```

pwm.setPWM(2, 0, 450)
pwm.setPWM(5, 0, 300)
pwm2.setPWM(14, 0, 270)
pwm2.setPWM(1, 0, 470)
time.sleep(w)

```

```

pwm.setPWM(0, 0, 345)
pwm.setPWM(1, 0, 490)
pwm.setPWM(3, 0, 360)
pwm.setPWM(4, 0, 390)
pwm.setPWM(12, 0, 260)
pwm.setPWM(13, 0, 350)
pwm.setPWM(15, 0, 450)
pwm2.setPWM(0, 0, 440)
time.sleep(w)

```

```

pwm.setPWM(2, 0, 500)
pwm.setPWM(5, 0, 250)
pwm2.setPWM(14, 0, 320)
pwm2.setPWM(1, 0, 420)
#time.sleep(0.5)

```

```

#####

```

```

pwm.setPWM(6, 0, 345)
pwm.setPWM(7, 0, 310)
pwm.setPWM(9, 0, 420)
pwm.setPWM(10, 0, 450)
pwm2.setPWM(2, 0, 325)
pwm2.setPWM(3, 0, 345)
pwm2.setPWM(5, 0, 485)
pwm2.setPWM(6, 0, 480)
time.sleep(w)

```

```

pwm.setPWM(8, 0, 340)
pwm.setPWM(11, 0, 250)
pwm2.setPWM(4, 0, 225)
pwm2.setPWM(7, 0, 475)
time.sleep(w)

```

```

pwm.setPWM(6, 0, 295)
pwm.setPWM(7, 0, 360)
pwm.setPWM(9, 0, 470)
pwm.setPWM(10, 0, 400)
pwm2.setPWM(2, 0, 275)
pwm2.setPWM(5, 0, 535)
pwm2.setPWM(3, 0, 395)
pwm2.setPWM(6, 0, 430)
time.sleep(w)

```

```

pwm.setPWM(8, 0, 290)
pwm.setPWM(11, 0, 300)
pwm2.setPWM(4, 0, 275)
pwm2.setPWM(7, 0, 425)
#time.sleep(0.5)
j = j + 1

```

```

def left2(n):

```

j = 0

while j < n:

```
pwm.setPWM(0, 0, 395)
pwm.setPWM(1, 0, 440)
pwm.setPWM(3, 0, 310)
pwm.setPWM(4, 0, 440)
pwm.setPWM(12, 0, 310)
pwm.setPWM(13, 0, 300)
pwm.setPWM(15, 0, 400)
pwm2.setPWM(0, 0, 490)
time.sleep(w)
```

```
pwm.setPWM(2, 0, 450)
pwm.setPWM(5, 0, 200)
pwm2.setPWM(14, 0, 270)
pwm2.setPWM(1, 0, 370)
time.sleep(w)
```

```
pwm.setPWM(0, 0, 345)
pwm.setPWM(1, 0, 490)
pwm.setPWM(3, 0, 360)
pwm.setPWM(4, 0, 390)
pwm.setPWM(12, 0, 260)
pwm.setPWM(13, 0, 350)
pwm.setPWM(15, 0, 450)
pwm2.setPWM(0, 0, 440)
time.sleep(w)
```

```
pwm.setPWM(2, 0, 500)
pwm.setPWM(5, 0, 250)
pwm2.setPWM(14, 0, 320)
pwm2.setPWM(1, 0, 420)
#time.sleep(0.5)
```

#####

```
pwm.setPWM(6, 0, 345)
pwm.setPWM(7, 0, 310)
pwm.setPWM(9, 0, 420)
pwm.setPWM(10, 0, 450)
pwm2.setPWM(2, 0, 325)
pwm2.setPWM(3, 0, 345)
pwm2.setPWM(5, 0, 485)
pwm2.setPWM(6, 0, 480)
time.sleep(w)
```

```
pwm.setPWM(8, 0, 340)
pwm.setPWM(11, 0, 350)
pwm2.setPWM(4, 0, 225)
pwm2.setPWM(7, 0, 375)
time.sleep(w)
```

```

pwm.setPWM(6, 0, 295)
pwm.setPWM(7, 0, 360)
pwm.setPWM(9, 0, 470)
pwm.setPWM(10, 0, 400)
pwm2.setPWM(2, 0, 275)
pwm2.setPWM(3, 0, 395)
pwm2.setPWM(5, 0, 535)
pwm2.setPWM(6, 0, 430)
time.sleep(w)

```

```

pwm.setPWM(8, 0, 290)
pwm.setPWM(11, 0, 300)
pwm2.setPWM(4, 0, 275)
pwm2.setPWM(7, 0, 425)
#time.sleep(0.5)
j = j + 1

```

```
def right2(n):
```

```

j = 0
while j < n:

```

```

    pwm.setPWM(0, 0, 395)
    pwm.setPWM(1, 0, 440)
    pwm.setPWM(3, 0, 310)
    pwm.setPWM(4, 0, 440)
    pwm.setPWM(12, 0, 310)
    pwm.setPWM(13, 0, 300)
    pwm.setPWM(15, 0, 400)
    pwm2.setPWM(0, 0, 490)
    time.sleep(w)

```

```

    pwm.setPWM(2, 0, 550)
    pwm.setPWM(5, 0, 300)
    pwm2.setPWM(14, 0, 370)
    pwm2.setPWM(1, 0, 470)
    time.sleep(w)

```

```

    pwm.setPWM(0, 0, 345)
    pwm.setPWM(1, 0, 490)
    pwm.setPWM(3, 0, 360)
    pwm.setPWM(4, 0, 390)
    pwm.setPWM(12, 0, 260)
    pwm.setPWM(13, 0, 350)
    pwm.setPWM(15, 0, 450)
    pwm2.setPWM(0, 0, 440)
    time.sleep(w)

```

```

    pwm.setPWM(2, 0, 500)
    pwm.setPWM(5, 0, 250)
    pwm2.setPWM(14, 0, 320)
    pwm2.setPWM(1, 0, 420)

```

```
#time.sleep(0.5)

#####

    pwm.setPWM(6, 0, 345)
    pwm.setPWM(7, 0, 310)
    pwm.setPWM(9, 0, 420)
    pwm.setPWM(10, 0, 450)
    pwm2.setPWM(2, 0, 325)
    pwm2.setPWM(3, 0, 345)
    pwm2.setPWM(5, 0, 485)
    pwm2.setPWM(6, 0, 480)
    time.sleep(w)

    pwm.setPWM(8, 0, 240)
    pwm.setPWM(11, 0, 250)
    pwm2.setPWM(4, 0, 325)
    pwm2.setPWM(7, 0, 475)
    time.sleep(w)

    pwm.setPWM(6, 0, 295)
    pwm.setPWM(7, 0, 360)
    pwm.setPWM(9, 0, 470)
    pwm.setPWM(10, 0, 400)
    pwm2.setPWM(2, 0, 275)
    pwm2.setPWM(3, 0, 395)
    pwm2.setPWM(5, 0, 535)
    pwm2.setPWM(6, 0, 430)
    time.sleep(w)

    pwm.setPWM(8, 0, 290)
    pwm.setPWM(11, 0, 300)
    pwm2.setPWM(4, 0, 275)
    pwm2.setPWM(7, 0, 425)
    #time.sleep(0.5)
    j = j + 1
```

Appendix B

The web server code: -

```
<!DOCTYPE html>

<html>
<head>
    <title>Spider Robot</title>
    <script type="text/javascript" src="/pico/pico.js"></script>
    <script type="text/javascript">
        pico.load('Spider');
        pico.main = function() {
            document.getElementById('forward').addEventListener('click', function(){
                Spider.forward(4);
            });
            document.getElementById('backward').addEventListener('click', function(){
                Spider.backward(3);
            });
            document.getElementById('left').addEventListener('click', function(){
                Spider.left(3);
            });
            document.getElementById('right').addEventListener('click', function(){
                Spider.right(3);
            });
            document.getElementById('standing').addEventListener('click',
function(){
                Spider.standing();
            });
            document.getElementById('initialize').addEventListener('click', function(){
                Spider.initialize();
            });
            document.getElementById('camup').addEventListener('click', function(){
                Spider.camera(1);
            });
            document.getElementById('camdown').addEventListener('click', function(){
                Spider.camera(2);
            });
            document.getElementById('camleft').addEventListener('click',
function(){
                Spider.camera(3);
            });
            document.getElementById('camright').addEventListener('click', function(){
                Spider.camera(4);
            });
        }
    </script>
</head>
<body background="/index_files/spider.jpg"><center>
    <h1>
        SPIDER ROBOT
    </h1>
```



```
<div>  
    <input type="image" id="camup" width="60" value="submit"  
src="./index_files/up.png">
```

```
<div>
    <input type="image" id="camleft" width="50"
value="submit" src="/index_files/left.png">
    <iframe src="http://192.168.1.125:8090/?action=stream"
width="350" height="280" align="center"></iframe>
    <input type="image" id="camright" width="50"
value="submit" src="/index_files/right.png">
</div>
```

```
<div>
    <button type="button" id="standing"
style="width:50px;height: 40px">STAND</button>
```

<button type="button" id="initialize"
style="width:50px;height: 40px">SIT</button>
</div>

 $\langle p \rangle$

</p><div>

```
</div>  
    <input type="image" id="left" width="60" value="submit"  
src="/index_files/spiderleft.png">  
        &nbsp;&nbsp; <input type="image" id="backward"  
width="60" value="submit" src="/index_files/backward.png">&nbsp;&nbsp; &nbsp;   <input type="image" id="right" width="60" value="submit"  
src="/index files/spiderright.png">
```

```
</div></center>
</body></html>
```

Appendix C

The code for server start-up: -

1. First created a file in /etc/init.d with a suitable name such as (vncboot, camboot or picoboot) with the following content.

```
# Description: Start (VNC, webcam, pico) Server at boot time.
```

```
#!/bin/sh
```

```
# /etc/init.d/(vncboot, camboot or picoboot)
```

```
USER=root
```

```
HOME=/root
```

```
export USER HOME
```

```
case "$1" in
```

```
start)
```

```
    echo "Starting (VNC, webcam, pico) Server"
```

```
    /usr/bin/vncserver :0 -geometry 1280x800 -depth 16 -pixelformat rgb565  
mjpg_streamer -i "/usr/lib/input_uvc.so -d /dev/video0 -r 320x240 -f 10" -o  
"/usr/lib/output_http.so -p 8090 -w /var/www/mjpg_streamer"
```

```
cd /root/Spider/Spider
```

```
python -m pico.server ;;
```

```
stop)
```

```
    echo "Stopping (VNC, webcam, pico) Server"
```

```
    /usr/bin/(VNC, webcam, pico)server -kill :0 ;;
```

```
*)
```

```
    echo "Usage: /etc/init.d/(vncboot, camboot or picoboot) {start/stop}"
```

```
    exit 1 ;;
```

```
esac
```

```
exit 0
```

2. Modify the file permissions so it can be executed

```
chmod 755 /etc/init.d/(vncboot, camboot or picoboot)
```

3. Enable dependency based boot sequencing

```
update-rc.d /etc/init.d/(vncboot, camboot or picoboot) defaults
```

4. if update-rc.d: error: unable to read /etc/init.d//etc/init.d/(vncboot, camboot or picoboot)

5. Then try the following command

```
update-rc.d (vncboot, camboot or picoboot) defaults
```

6. Reboot your Raspberry PI and you should find a (VNC, webcam, pico) server already started.

Appendix D

The making of the chopstick spider robot: -

Parts and Tools Required

Raspberry Pi (1)
Keyboard & Mouse (1)
HDMI cable (1)
Wi-Fi dongle compactible with raspberry pi (1)
Any Webcam compactible with raspberry pi (1)
I2c 32 channel PWM Servo Board (1)
Powered USB hub (1)
12Kg/cm servo with metal gears (8)
5Kg/cm servo (16)
Self-tapping screws Small pan head 2 x 6mm (200)
Self-tapping screws Small pan head 2 x 8mm (50)
Self tapping screws 2.3mm x 12mm (20)
Miniature servos (2)
Polymorph granules (500g)
Disposable Chopsticks Bamboo (40+)
5V Rechargeable battery pack 2000mAH or better (1)
11.1V Rechargeable battery pack 1800mAH or better (1)
Cable ties heavy duty 200mm long (4)
Cable ties light duty, 100mm long (24+)
Switch (2)
Phillips Screwdriver (1)
Kettle
Side cutlers

Step 1: Prepare your workspace.



- A glass-top table is best when using Polymorph. You need a flat workspace that the Polymorph won't stick to. If you have a wood table then keep the top wet or put a sheet of glass on top.
- You will need a rag, a constant supply of hot water, a bowl for the hot water (not plastic as the Polymorph may bond to it) and a cup of cold water (ice will speed things up) for cooling the Polymorph. Coffee is optional. :)
- When you dip hot Polymorph in cold water for a few seconds the outside will cool rapidly forming a milky, flexible skin. This will prevent the Polymorph from bonding to your servos. The exact amount of time required will depend on how hot your Polymorph is and how cold the water is. A slightly milky appearance is your best indicator.
- For each leg segment I used a 230mm chopstick cut in half, each segment being about 115mm in length. Cut 16 chopsticks in half with the side cutters now so they are on hand when the Polymorph is hot.

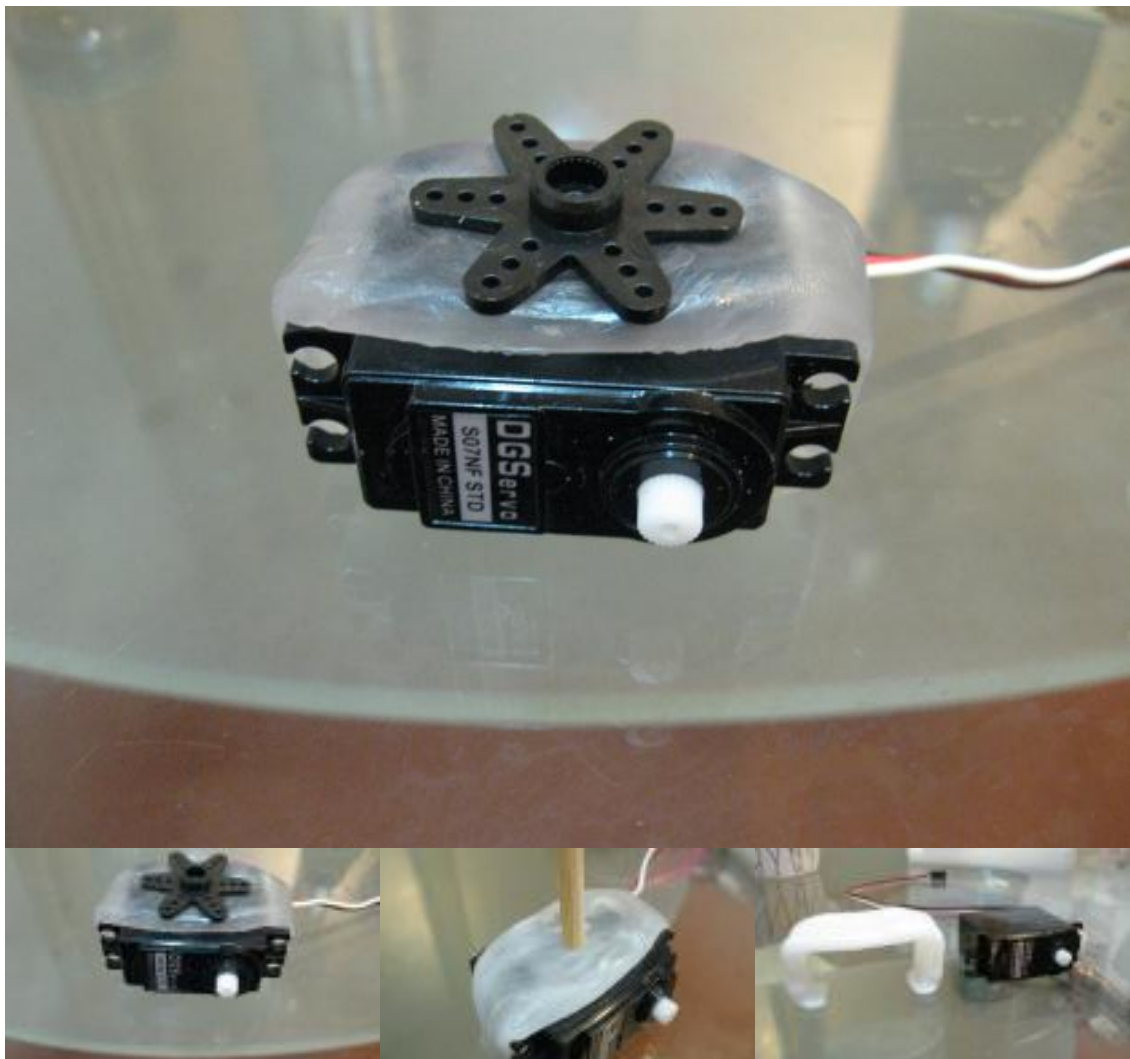
- I have used pan-head self-tapping screws for this project. Polymorph is perfect for self-tapping screws and the pan heads eliminate the need for washers.

Step 2: Prepare your Polymorph.



- Polymorph is actually Polycaprolactone, a non-toxic, re-usable, biodegradable plastic. You can read about it [here](#).
- Boiling water can be dangerous. This project should not be attempted by children without adult supervision.
- Put in only as much Polymorph as you need. If you put in too much at once you can end up with a big lump that is slow to reheat.
- When you put the Polymorph granules into the hot water they will quickly become transparent and sticky. Once all the granules are transparent use a chopstick (because the water is very hot) to remove some from the bowl. Squeeze it gently in your hands to remove water trapped inside.
- The Polymorph is now a putty that can be easily shaped by hand. At this time it will also stick to objects so be careful where you put it. It can bond to the plastic servo housing if it is too hot. As it cools it will slowly harden and become white again. When completely cooled it is a plastic similar to nylon.

Step 3: Making a hip joint.



- The first piece to make is a cradle for the thigh servo. This piece allows the thigh servo to be attached to the hip servo. As the robot can be quite heavy this part needs to be quite strong. The thigh servo clips into this piece tightly with 4 mounting screws ensuring the servo cannot unclip.
- Roll up a ball of Polymorph about 30-35mm in diameter. Then roll the ball into a thick rod about 40mm long. Dip this in the cold water for a few seconds to ensure that the Polymorph does not bond to your servo.
- Wrap this rod around your servo as shown in the photo making sure you have enough Polymorph around the servo mounting holes for the mounting screws.
- Press your servo horn gently into the Polymorph making sure it is reasonably well centered. Do not press too hard as you need at least 3-4mm of Polymorph behind the servo horn for the mounting screws to go into.
- As the Polymorph cools, remove the servo horn and poke a hole about 6mm in diameter in the center of the servo horn impression. This will allow you to insert the screw that holds the servo horn to the servo.

- Once it has fully cooled you should be able to unclip the servo. If your Polymorph was too hot and bonded to the servo then use a flat-blade screwdriver or a knife and gently pry the Polymorph away from the servo.
- Mount your servo horn with 2x6mm pan-head screws. Attach the servo cradle to the hip servo and align it so that when the hip servo is in the center position the thigh will be perpendicular to the body.

Step 4: Making a thigh - mount the knee servo.

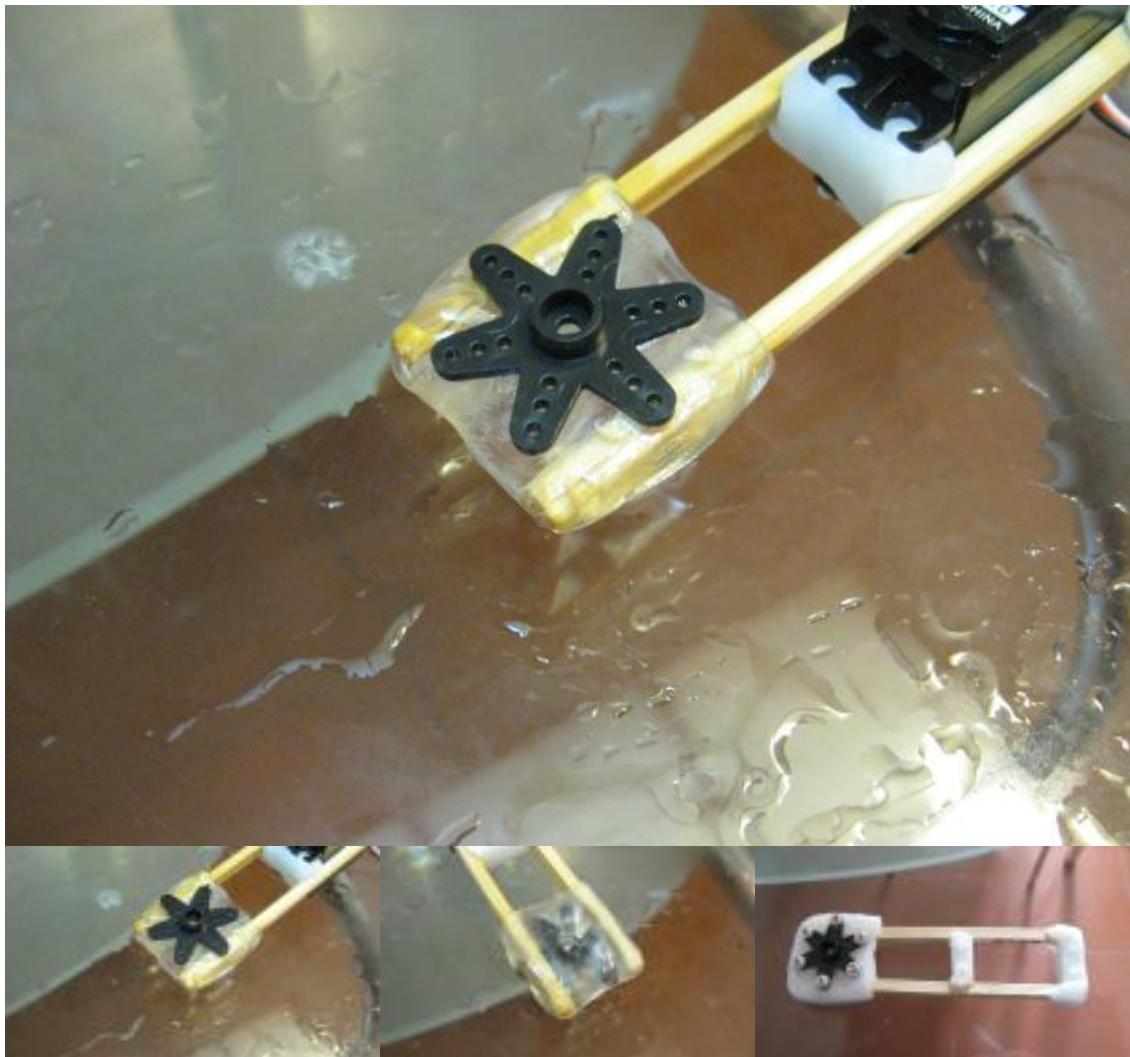


- The next piece is the thigh. The chopsticks are usually tapered with a thick end to be held and a thin end for holding the food. Use two thick sections for the thigh. We will use the thinner sections for the legs. My chopsticks are square at the thick end and round at the thin end but the shape is not important.
- Start with a small ball of Polymorph about 10-15mm in diameter and roll this into a thick rod about 40mm long. Press your chopsticks into the ends of the Polymorph rod while it is very hot so they bond. Press the Polymorph fully around the ends of the chopsticks so it covers about 10mm of the chopsticks.
- Dip the Polymorph into the cold water for a few seconds and then place a 5Kg/cm knee servo between the chopsticks. Shape your Polymorph so that the

servo is held firmly between the chopsticks with a solid section for the servo mounting screws to dig into.

- Hold everything in place for a few minutes while the Polymorph cools and hardens. Once it is completely cool the servo should be a tight fit between the chopsticks but easily removed. Insert two 8x2mm mounting screws to hold the servo in place.
- To make the small servo mount for the other end, get a small ball of Polymorph about 10-12mm in diameter and roll it into a thick rod about 20mm long. Dip this in cold water for a few seconds. Now press it into the other servo mount and make sure it goes around the chopsticks enough so that when cool it can slide along the length of them without falling out. This makes it easy to remove the servo if you need to replace it.
- Once the Polymorph has hardened use two 8x2mm pan-head screws to fix the servo to the new servo mount.

Step 5: Making a thigh - the thigh joint.



1. Now get a big ball of Polymorph about 25mm in diameter. Squash it flat and wrap it around the other end of the chopsticks while the Polymorph is very hot. You should have a flat area big enough to mount another servo horn.
2. Dip the Polymorph in cold water for a few seconds and then gently press a servo horn into the Polymorph so it sits flat and makes an impression in the Polymorph. While the Polymorph is soft, create a 6mm-diameter hole in the center for the servo horn mounting screw to go through.
3. Once the Polymorph has cooled mount the servo horn using 2x6 pan-head screws. Make sure the hole for the servo horn mounting screw lines up with the center of the servo horn.
4. Your thigh segment is now complete and can be mounted on the thigh servo. Align your thigh so that when the thigh servo is in center position the thigh is parallel to the ground.

Step 6: Making a leg.

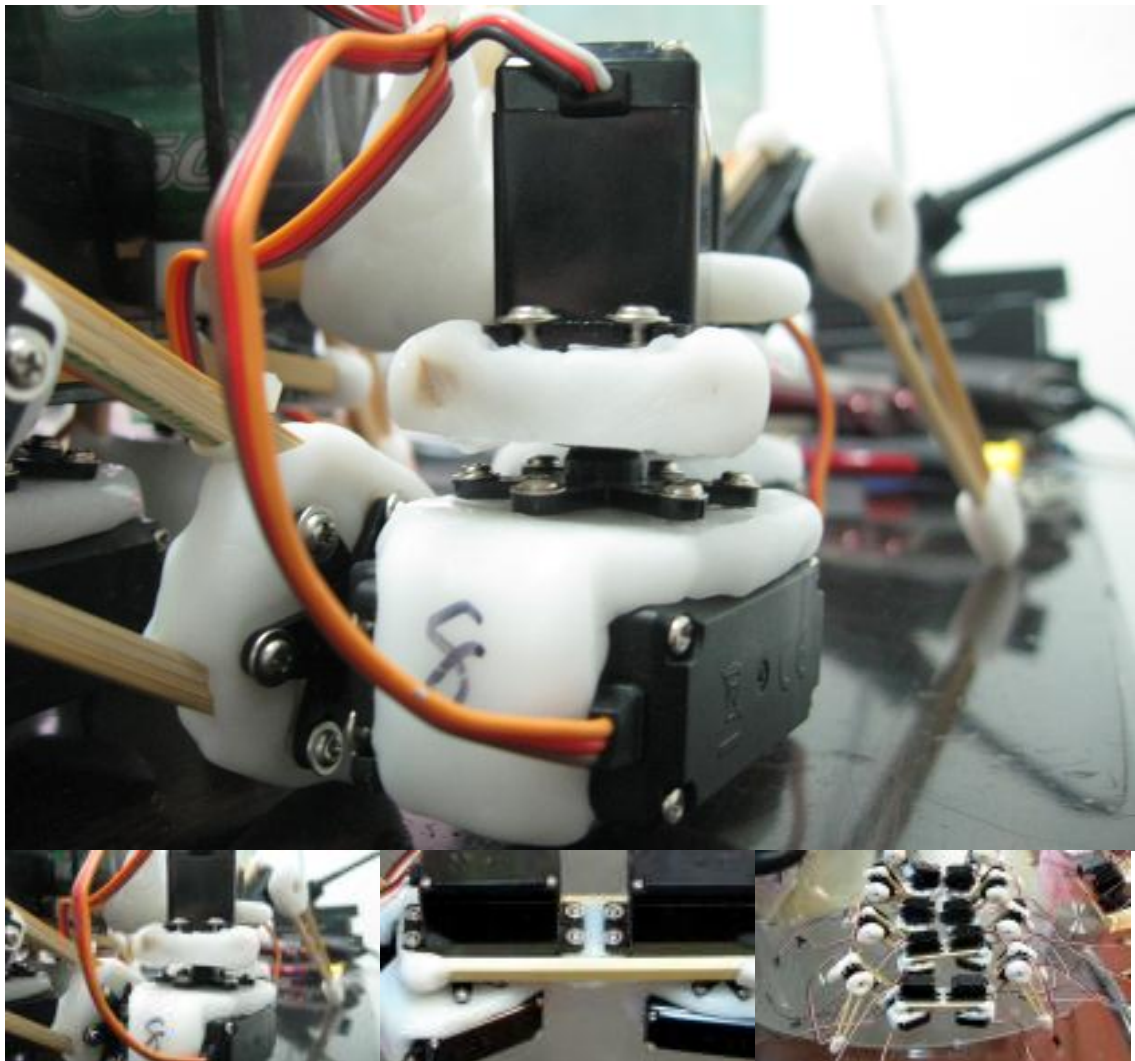


- The leg is made using two of the thinner halves of the chopstick. Get a ball of Polymorph about 25mm in diameter and poke your two chopsticks into it, one

on either side, while it is very hot. Knead the Polymorph around the chopsticks and hold the other ends of the chopsticks together to form a "V" shape.

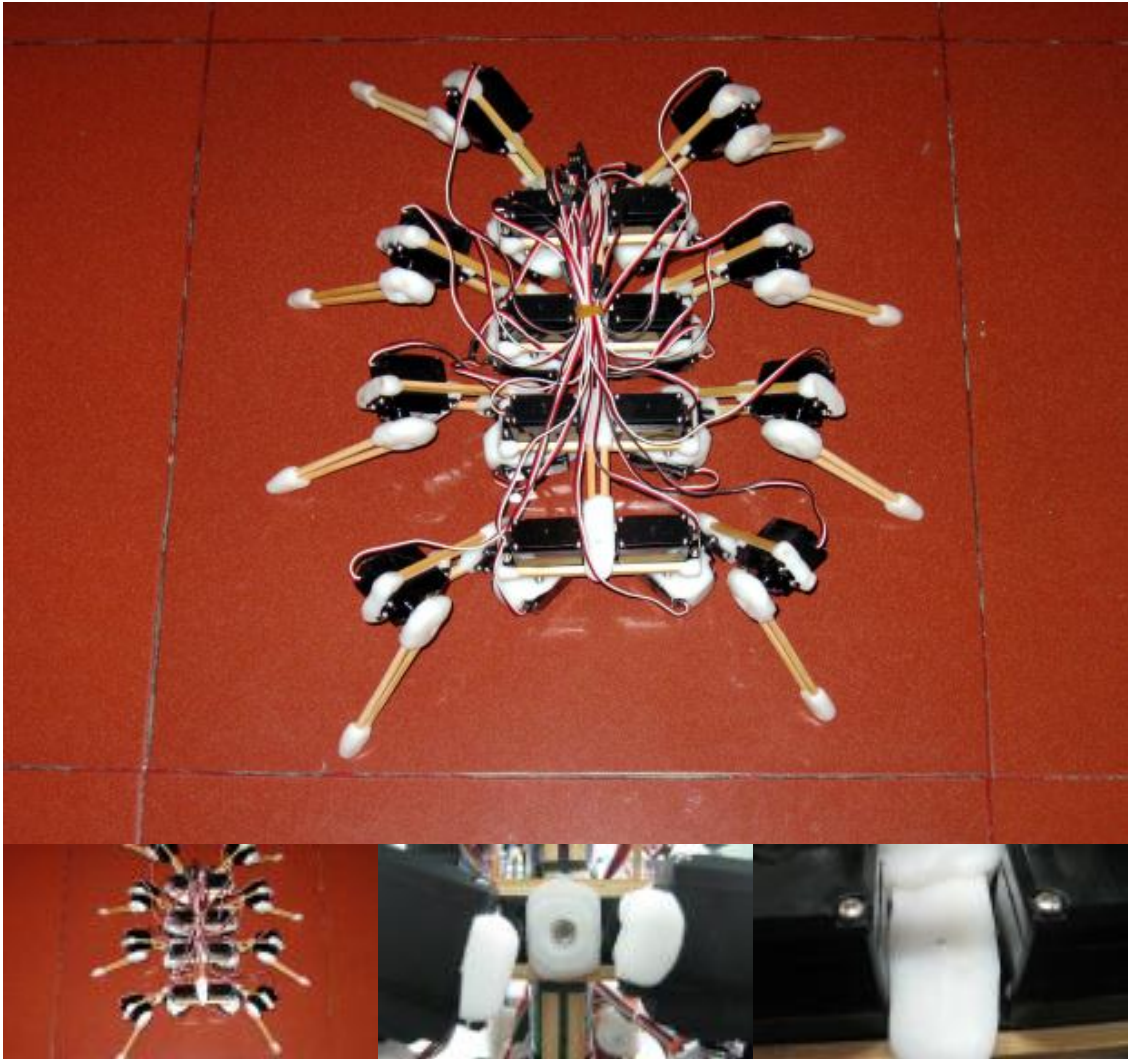
- Wet a spot on the table so that the Polymorph won't stick and then squash your ball into a thick disc on the table. Poke a 6mm-diameter hole in the center of the disc for your servo horn mounting screw.
- While the Polymorph is still soft, gently press a servo horn into the Polymorph to leave an impression. Make sure the hole in the center lines up.
- While the top of the leg is cooling get a small ball of hot Polymorph and squash it around the other end of the chopsticks to make a foot.
- Once the leg has cooled, mount the servo horn with 2x6mm pan-head screws. Mount the leg so that when the knee servo is in its center position the leg is at 90 degrees to the thigh.
- Repeat steps 3-6 until you have as many legs as required. Remember to make left and right legs in equal numbers.

Step 7: Joining your legs into pairs.



- Join your left and right legs together in pairs. Start by making another ball of Polymorph about 15-20mm in diameter. Roll it into a short rod about 40mm long.
- While the Polymorph is very hot, press a thick section of chopstick into each end and knead the Polymorph so it covers about 10mm of each chopstick.
- Dip the Polymorph in cold water for a few seconds and then place the chopsticks either side of a left hip servo. Note this time I've put the Polymorph on the other side of the servo mount. This allows my mounting screws to be inserted from the top of the robot instead of the bottom.
- Make sure the chopsticks are pressed tight against the sides of the servo and there is plenty of Polymorph for the servo mounting screws to dig into. Also check your clearance so that the hip can swing freely.
- Once the Polymorph has cooled, use 2x8mm pan-head screws to secure the hip servo. Repeat the process at the other end with the right hip servo.
- There should be a gap between the left and right hip servos in the center of the frame you just made. Roll up a ball about 20mm in diameter and dip it in cold water for a few seconds. Press it into the center between the two servos.
- While this center piece is cooling, make an impression in the center with your finger so the Polymorph is only about 4mm thick in the center. Use a skewer to poke a small hole about 3mm in diameter through the center for a mounting screw to go through.

Step 8: Add a spine.

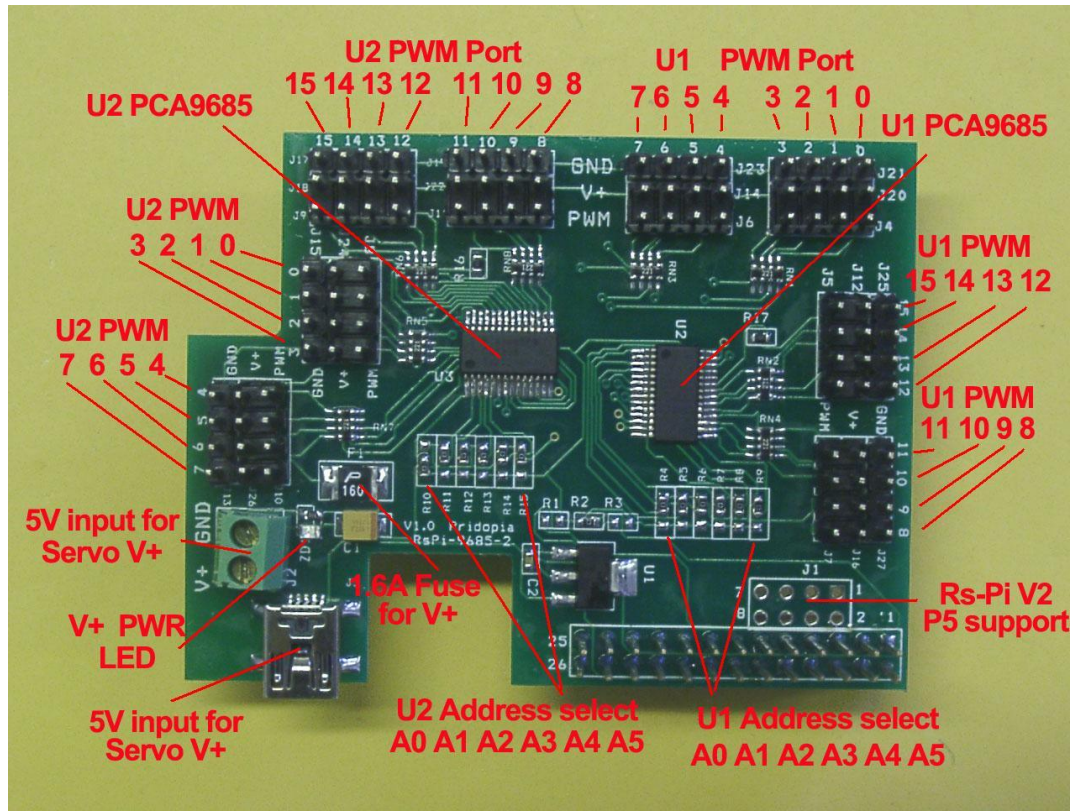


- A spine can be made from two whole chopsticks. Roll up a hot ball of Polymorph about 20-25mm in diameter and press your two chopsticks into it. Knead your Polymorph around the sticks so they are parallel with a gap of about 3-4mm in the center. Dip it in cold water for a few seconds and check that it fits between the servo pairs. Adjust your center gap for a neat fit.
- Do the same at the other end so that your chopsticks remain parallel. Wet the table top slightly so the Polymorph doesn't stick and press your spine gently on the table top to ensure it is completely flat when the Polymorph hardens. You may wish to extend the ends slightly so you have a section of Polymorph where you can mount sensors, switches etc.
- Make sure each ball is hot so it bonds with the chopsticks and knead it evenly around them. Dip it in cold water and then check that it fits between the left and right hip servos.
- Once again wet the table top and press the spine gently against the table top so that the spine sits completely flat on the table while the center pieces cool.
- Use 2.3x12mm screws to fix your leg pairs to your spine.

Appendix E

The product details of the interface board: -

Raspberry Pi - I2C 32 Channel 12 bit PWM Servo Board



32 channel PWM / Servo

The PCA9685 is an I2C-bus controlled 16-channel LED controller optimized for LCD

Red/Green/Blue/Amber (RGBA) color backlighting applications. Each LED output has its

own 12-bit resolution (4096 steps) fixed frequency individual PWM controller that operates

at a programmable frequency from a typical of 40 Hz to 1000 Hz with a duty cycle that is

adjustable from 0 % to 100 % to allow the LED to be set to a specific brightness value.

All outputs are set to the same PWM frequency.

PCA9685 also has a built-in oscillator for the PWM control.

However, the frequency used for PWM control in the PCA9685 is adjustable from about 40 Hz to 1000 Hz as compared to the typical 97.6 kHz frequency of the PCA9635. This allows the use of PCA9685 with external power supply controllers.

All bits are set at the same frequency.

The PCA9685 has 4096 steps (12-bit PWM)

1. J3 Mini USB 5V input for PWM V+
- J2 2P Terminal Block 5V input for PWM V+
2. J1 Rs-Pi V2 GPIO output
3. U2 PCA9685 (PWM Port 0 ~ 15)
4. R4,R5,R6,R7,R8,R9(for U2 Address select A0,A1,A2,A3,A4,A5)
5. U3 PCA9685 (PWM Port 0 ~ 15)
6. R10,R11,R12,R13,R14,R15(for U3 Address select A0,A1,A2,A3,A4,A5)
- 7.Red power-good V+ LED
8. 1.6A PolySwitch Fuse for V+ input protect.

```
COM34 - PuTTY
Adafruit_I2C.pyc  Adafruit_PWM_Servo_Driver.pyc  Servo_Example
root@raspberrypi:/home/pi/pwm# i2cdetect -y 0
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: 40 41 -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: 70 -- -- -- -- -- -- -- -- -- -- -- -- -- --
root@raspberrypi:/home/pi/pwm# dir
Adafruit_I2C.py  Adafruit_PWM_Servo_Driver.py  servo-41.py
o.py
Adafruit_I2C.pyc  Adafruit_PWM_Servo_Driver.pyc  Servo_Example
root@raspberrypi:/home/pi/pwm# ./servo-41.py
Resetting PCA9685
Setting PWM frequency to 60 Hz
Estimated pre-scale: 100
Final pre-scale: 101
```

I2Cdetect you can found 2 device in system (40,41)