BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT

THÀNH PHỐ HỒ CHÍ MINH

ĐỒ ÁN TỐT NGHIỆP

NGÀNH CNKT ĐIỀU KHIỂN VÀ TỰ ĐỘNG HÓA

DESIGN, IMPLEMENTATION AND CONTROL OF HEXAPOD ROBOT COMBINING IMAGE PROCESSING ON ANDROID PLATFORM

GVHD: NGUYEN VAN THAI, PhD

SVTH: PHUNG TU MINH

MSSV: 16151028

SVTH: NGUYEN HOANG HUYNH

MSSV: 16151048

Tp. Hồ Chí Minh, tháng 07/2020

HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY AND DUCATION

FACULTY FOR HIGH QUALITY TRAINING

GRADUATION PROJECT

DESIGN, IMPLEMENTATION AND CONTROL OF HEXAPOD ROBOT COMBINING IMAGE PROCESSING ON ANDROID PLATFORM

PHUNG TU MINH

Student ID: 16151028

NGUYEN HOANG HUYNH

Student ID: 16151048

Major: AUTOMATIC AND CONTROL ENGINEERING TECHNOLOGY

Advisor: NGUYEN VAN THAI, PhD.

Ho Chi Minh City, July 2020

THE SOCIALIST REPUBLIC OF VIETNAM
Independence – Freedom– Happiness

Ho Chi Minh City, August 4, 2020

GRADUATION PROJECT ASSIGNMENT

Student name: Phung Tu Minh Student ID: 16151048

Student name: Nguyen Hoang Huynh Student ID: 16151028

Major: Automation And Control Engineering Technology Class: 16151CL3

Advisor: PhD. NGUYEN VAN THAI Phone number: 0902.807.576

Date of assignment: 24/02/2020 Date of submission: 26/07/2020

1. Project title: Design, Implementation and Control of Hexapod Robot combining image processing on Android platform

2. Initial materials provided by the advisor: _____

3. Content of the project: _____

4. Final product: _____

CHAIR OF THE PROGRAM
(Sign with full name)

ADVISOR
(Sign with full name)

THE SOCIALIST REPUBLIC OF VIETNAM
Independence – Freedom– Happiness

Ho Chi Minh City, August 4, 2020

ADVISOR'S EVALUATION SHEET

Student name: Phung Tu Minh Student ID: 16151048

Student name: Nguyen Hoang Huynh Student ID: 16151028

Major: Automation And Control Engineering Technology

Project title: Design, Implementation and Control of Hexapod Robot combining image processing on Android platform.

Advisor: PhD. NGUYEN VAN THAI

EVALUATION

1. Content of the project:

.....

.....

.....

2. Strengths:

.....

.....

.....

3. Weaknesses:

.....

.....

.....

4. Approval for oral defense? *(Approved or denied)*

.....

5. Overall evaluation: (Excellent, Good, Fair, Poor)

.....

6. Mark:…….(*in words*:.....)

Ho Chi Minh City, August 4, 2020.

ADVISOR

(Sign with full name)

THE SOCIALIST REPUBLIC OF VIETNAM
Independence – Freedom– Happiness

Ho Chi Minh City, August 3, 2020

PRE-DEFENSE EVALUATION SHEET

Student name: Phung Tu Minh Student ID: 16151048

Student name: Nguyen Hoang Huynh Student ID: 16151028

Major: Automation And Control Engineering Technology

Project title: Design, Implementation and Control of Hexapod Robot combining image processing on Android platform.

Name of Reviewer: PhD. Dang Xuan Ba.

EVALUATION

1. Content and workload of the project

   The project is a combination of many technologies such as image processing, robot, and web application.

2. Strengths:

   It could be applied to real-life missions.

3. Weaknesses:

   The project seems to be a challenge for students in tutoring the new technologies rather than adopting what they learned for an intensive applications. As a result, the robot could be controllable, yet in poor performance. Its advantages over wheel robots have not been shown or tested. Effectiveness of tracking control and the management system should be shown more clearly.

4. Approval for oral defense? *(Approved or denied)*

   Approved

5. Overall evaluation: *(Excellent, Good, Fair, Poor)*

   Good

6. Mark: 8.5 (*in words*: Eight point five)

Ho Chi Minh City, August 3, 2020.

REVIEWER

(Sign with full name)

THE SOCIALIST REPUBLIC OF VIETNAM

Independence – Freedom– Happiness

Ho Chi Minh City, August 4, 2020

EVALUATION SHEET OF DEFENSE COMMITTEE MEMBER

Student name: Phung Tu Minh Student ID: 16151048

Student name: Nguyen Hoang Huynh Student ID: 16151028

Major: Automation And Control Engineering Technology

Project title: Design, Implementation and Control of Hexapod Robot combining image processing on Android platform.

Name of Defense Committee Member:

.....

EVALUATION

1. Content and workload of the project

.....

.....

.....

.....

2. Strengths:

.....

.....

.....

3. Weaknesses:

.....

.....

.....

4. Overall evaluation: *(Excellent, Good, Fair, Poor)*

.....

5. Mark:…….(*in words*:.....)

Ho Chi Minh City, August 4, 2020.

COMMITTEE MEMBER

(Sign with full name)

DISCLAIMER

We assure you that this is our research project and is guided science by PhD. Nguyen Van Thai. The research content, the results in this topic are honest. The figures in the table for analysis, evaluation and comments were collected by the authors from different sources specified in the references.

ACKNOWLEDGEMENTS

It is a great pleasure to thank all of our teachers, friends that supported us in the process of doing our thesis.

We're very glad to express our gratitude towards Nguyen Van Thai, PhD for his time and effort in helping us to learn much more scientific concepts, basic and advance knowledge. For your right directions that we can complete this project. He has given us the enthusiasm and creativity, in order to be continuous innovation to develop our thesis. We always remember the time when we first met him at 3DVisionLab to discuss about the thesis.

Many thanks to our vice president of HCMC University of Technology and Education, Ngo Van Thuyen, PhD who is in charge of scientific research, international relations, public relations, construction of ODA projects, deploy KPIs and quality assurance of our university. We have the opportunity to learn his subject - SCADA, thanks to his rigor and meticulousness, we have learned not only the knowledge but also how an engineer must become.

I would like to thank Nguyen Minh Tam, PhD Dean of Faculty of Electrical and Electronics Engineering and Vu Van Phong, PhD. They never hesitate to help us when we encounter problems. And we could not have done the research without all opportunities which Le My Ha, PhD and MS. Nguyen Tran Minh Nguyet, from Department of Automation Control of Faculty of High Quality Training, created for us. Especially, M.Eng Nguyen Tran Minh Nguyet was very closed to us, we learnt really a lot from our discussions and her encouragement as well as life lessons of Le My Ha, PhD.

Last but not least, we would like to express our deepest thanks to parents and families for sticking to as well as supporting us during these 4 years of difficult and challenging university. It is the belief and heartiness of the family that has motivated us to complete the final thesis. We will accomplish this project outright and fulfill the promise of success with our family.

TABLE OF CONTENT

LIST OF FIGURE

LIST OF TABLE

ABSTRACT

The thesis "Design, Implementation and Control of Hexapod Robot combining image processing on Android platform" was developed based on the idea of building a robot with flexibility in movement as well as feasibility in transportation, merchandise transference, rescue and exploration. Robot is designed with features that allow to be controlled even from short and long distances, balance terrain response, and process follow-up objects, namely circular-shaped objects in custom applications included tracking, monitoring and directions.

To do that, robot is applied dynamic algorithms to control the structure and gait. Thanks to the kinetic algorithm, the robot is able to respond to basic movements like a living creature with the flexibility of a 6-leg mechanism. By using the mobile phone itself as a virtual eyes, the robot will actually observe and recognize objects through image processing.

This thesis will present the overview in process of building robot's hardware and software, the design of our team to build a complete structure with 3 important goals: Stability, Easy to approach, Meet practical requirements. (SEM).

TÓM TẮT

Luận văn “Thiết kế và lập trình điều khiển Robot 6 chân kết hợp xử lý ảnh trên nền tảng Android” được phát triển dựa trên ý tưởng xây dựng một Robot có sự linh hoạt trong di chuyển cũng như khả thi trong các nhiệm vụ vận chuyển, cứu hộ và thăm dò. Robot được thiết kế với các tính năng điều khiển trong khoảng cách gần và khoảng cách xa, tính năng cân bằng đáp ứng địa hình, và xử lý ảnh bám theo vật, cụ thể là các vật có dạng hình tròn trong các ứng dụng theo dõi, giám sát và chỉ đường.

Để làm được điều đó, Robot được áp dụng các giải thuật động học nhằm điều khiển cơ cấu và dáng di chuyển. Nhờ có giải thuật động học, robot có thể đáp ứng các di chuyển cơ bản như một sinh vật sống với sự linh hoạt của cơ cấu 6 chân. Bằng việc sử dụng chính chiếc điện thoại di động như một mắt ảo, robot sẽ thật sự quan sát và nhận biết vật thể thông qua xử lý ảnh.

Luận văn này sẽ trình bày những nội dung tổng quan về quá trình xây dựng phần cứng và phần mềm của Robot, những thiết kế của nhóm để xây dựng một cơ cấu hoàn chỉnh với 3 mục tiêu quan trọng: Ổn định, dễ dàng tiếp cận và đáp ứng yêu cầu thực tế.

KEY WORD

PWM	**P**ulse-**w**idth **m**odulation
MPU	**M**icro **P**rocessor **U**nit
SPI	**S**erial **P**eripheral **B**us
SDA	**S**erial **D**ata
SCL	**S**erial **C**lock
GY	**Gy**roscope
IDE	**I**ntegrated **D**evelopment **E**nvironment
RGB	**R**ed - **G**reen - **B**lue
HUE	**H**ue - **S**aturation - **V**alue
API	**A**pplication **P**rogramming **I**nterface

CHAPTER 1 GENERAL

1.1 INTRODUCTION

Monitoring the chemical, nuclear and mobility environments in those environments for normal robots is impossible. Using a wireless robot to monitor and assess the surroundings helps to solve the given problem safely. Moving through rough surfaces, working independently in unsafe environments is a remarkable advantage that helps operators to be safe and regularly receive information and status of the environment.

1.2 OBJECTIVES OF THE THESIS

The data collection and monitoring from robots has made the operator convenient and fast. And the next step is that the robot can process the information received from the environment and process the information as programmed. To meet the requirements set by the team has planned development for spider robots. So the team chose the thesis "Design, Implementation and Control of Hexapod Robot combining image processing on Android platform".

- The issues are being studied for the topic.
- Study on positive and negative kinetic methods of robots.
- Robot motion trajectory.
- Research image processing recognition - tracking objects on the Android platform.
- App Design on Android to control robots.

1.3 LIMITATION OF TOPIC

A spider robot is designed to consist of 6 legs and each leg consists of 3 joints. So 1 foot will use 3 RC Servo MG996R motors. The limited rotation angle of the RC Servo motor is from 0 to 180 degrees. So the rotation angle will be limited. The total number of servos used for the 18-motor robot so the robot weight is about 8kg, so the robot will move quite slowly to ensure the system is immediately depressed.

1.4 THE CONTENT OF THESIS

The topic will be presented in parts from theoretical application to application. Each issue in the article will be presented through each chapter as follows:

Chapter 1: GENERAL

Chapter 2: THEORITICAL BASIC

Chapter 3: THE APP APPLICATION DESIGN AND HARDWARE DESIGN

Chapter 4: THE ALGORITHM

Chapter 5: THE EXPERIMENT RESULT/FINDING AND ANALYSIS

Chapter 6: CONCLUSION AND RECOMMENDATIONS

CHAPTER 2 THEORITICAL BASIC

2.1 KINEMATIC ANALYSIS

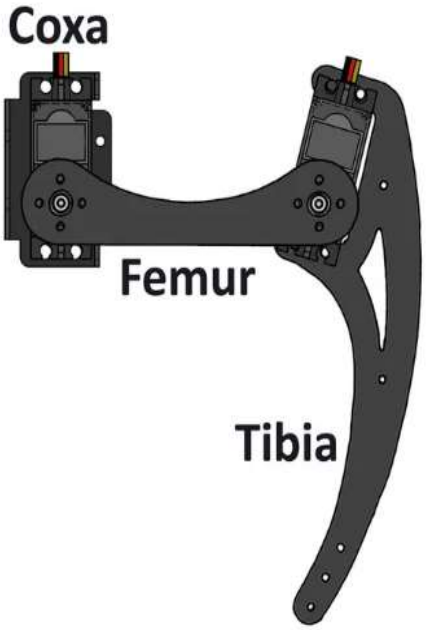


Figure 2. 1: The leg segment model.

The leg segment closest to the body will be called the coxa. In our build, the coxa is a servo bracket assembly we will start building in the next step. The second leg segment is the femur. The third leg segment, the last one on each leg, is the tibia.

The servos will be named according to the leg segment they move. The servo that attaches to the body of the hexapod is the coxa servo. The servo that lifts and lowers each leg is the femur servo. The servo that allows the legs to bend in the middle is the tibia servo.

Table 2. 1: Specification for hexapod robot.

Body parameter	**Dimension**
Body length	180mm
Body width	140mm
Body height	52mm
Body weight	2200g
Leg parameter	**Dimension**
Coxa length(L1)	30mm
Femur length(L2)	70mm
Tibia length(L3)	110mm
Coxa mass	2260g
Femur mass	105g
Tibia mass	57g

A hexapod is a kid of walking robot that has a high redundancy degree of freedom (DOF), coupling multi-branched, combine of series and parallel.

The hexapod has six legs, Each leg has three DOFs, so six legs have 18 DOFs, The kinematic of a hexapod consist of joint data of the eighteen legs, the positions at the end of legs, and the mathematic between trunk position and trunk poster.

This hexapod's structure is based on bionic. Each leg has three DOFs. On the left side of the trunk has three legs and that three legs on the right side . This robot's structure is based on insect as illustrated in figure 2.2.

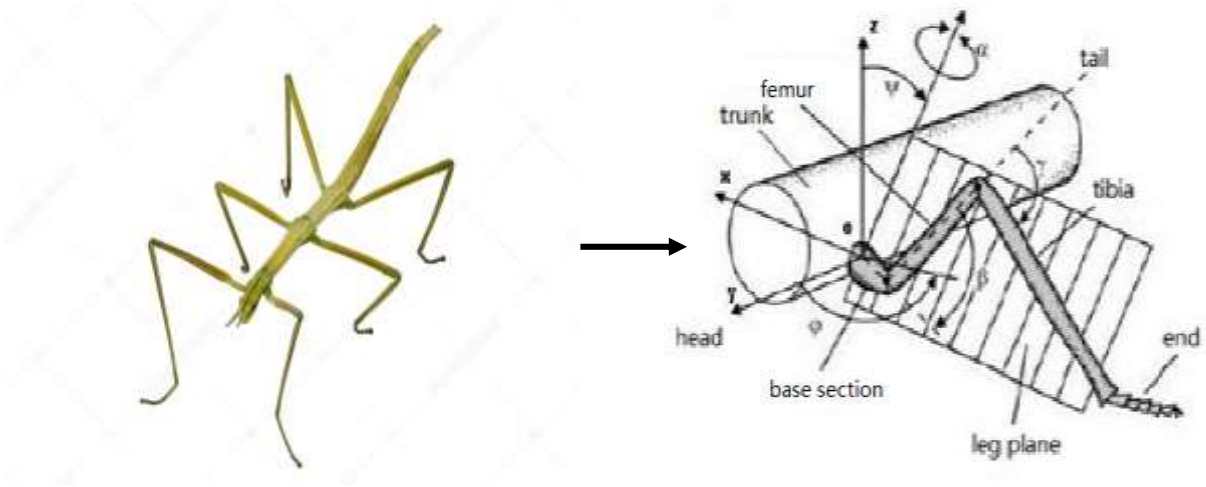


Figure 2. 2: Insect model.

In the figure 2.2, we have x is the coordinate axis along the side of the body, y is the coordinate axis along the direction of straight motion, z is the coordinate axis pointing up the body. Use the right-hand rule for axis definition. The alpha angle is the angle of rotation between body and base. The beta angle is the angle of rotation between the base section and the femur. The alpha angle is the angle between the base section and tibia .

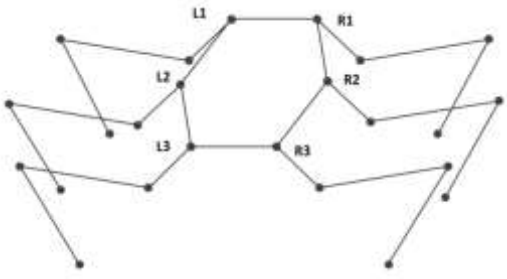


Figure 2. 3: Leg model of hexapod.

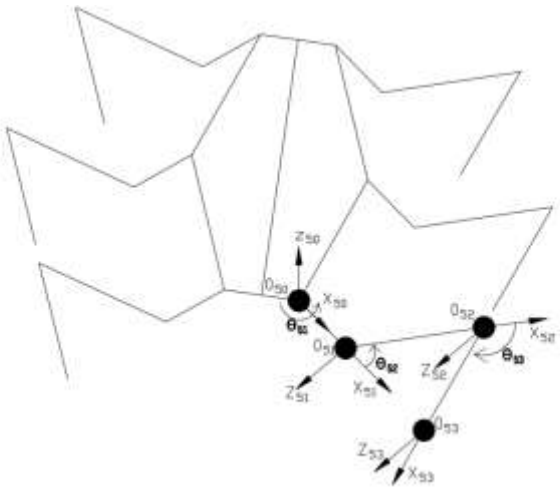


Figure 2. 4: Single leg coordinate.

In the figure 2.4 R3 leg position is marked by the coordinate system of the rotary joint and the foot position of the robot. Table D-H will be established from matrix conversion as seen in equation 2.1[1].

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.\ 1) $$

The leg structure of the robot is the same and parallel string, so making a D-H table for a single leg will apply to the related legs.

Using parameters in table 2.2 for the coordinate system in figure 2.4, we calculate the conversion matrix for each link, substitute parameters in table 2.2 into equation 2.1, we obtained [2]:

Table 2. 2: The D-H matrix parameter of hexapod robot.

Joint	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	z	θ_1
2	90^o	l1	0	θ_2
3	0	l2	0	θ_3
4	0	l3	0	0

The D-H parameters are the angles of two link , link distance ,link length , and the twisting angle of link .

$$ {}^{0}_{1}T = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.\ 2) $$

$$ {}^{1}_{2}T = \begin{bmatrix} c_2 & s_2 & 0 & l1 \\ 0 & 0 & -1 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.\ 3) $$

$$ {}^{2}_{3}T = \begin{bmatrix} c_3 & -s_3 & 0 & l2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.\ 4) $$

Conversion matrix of the entire coordinate system of R3-leg:

$$ {}^{0}_{3}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T $$

Obtain:

$$ \begin{bmatrix} c_{23}*c_1 & -s_{23}*c_1 & s_1 & c_1*(l1+l2*c_2) \\ c_{23}*s_1 & -s_{23}*s_1 & -c_1 & s_1*(l1+l2*c_2) \\ s_{23} & c_{23} & 0 & z+l2*s2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.\ 5) $$

We have $c_1, s_1, c_2, s_2, c_3, s_3, c_{23}, s_{23}$ are stand for :

$\cos(\theta_1), \sin(\theta_1), \cos(\theta_2), \sin(\theta_2), \cos(\theta_3), \sin(\theta_3), \cos(\theta_2+\theta_3), \sin(\theta_2+\theta_3)$

Assume $\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$ is the endpoint coordinates of leg-R3 in contact with the moving surface, multiply two equation 2.5 and equation 2.6 :

$$ \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} = {}^{0}_{3}T * \begin{bmatrix} l3 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} c_1*(l1+l3*c_{23}+l2*c_2) \\ s_1*(l1+l3*c_{23}+l2*c_2) \\ z+l3*s_{23}+l2*s_2 \\ 1 \end{bmatrix} \qquad (2.\ 6) $$

2.2 INVERSE KINEMATIC

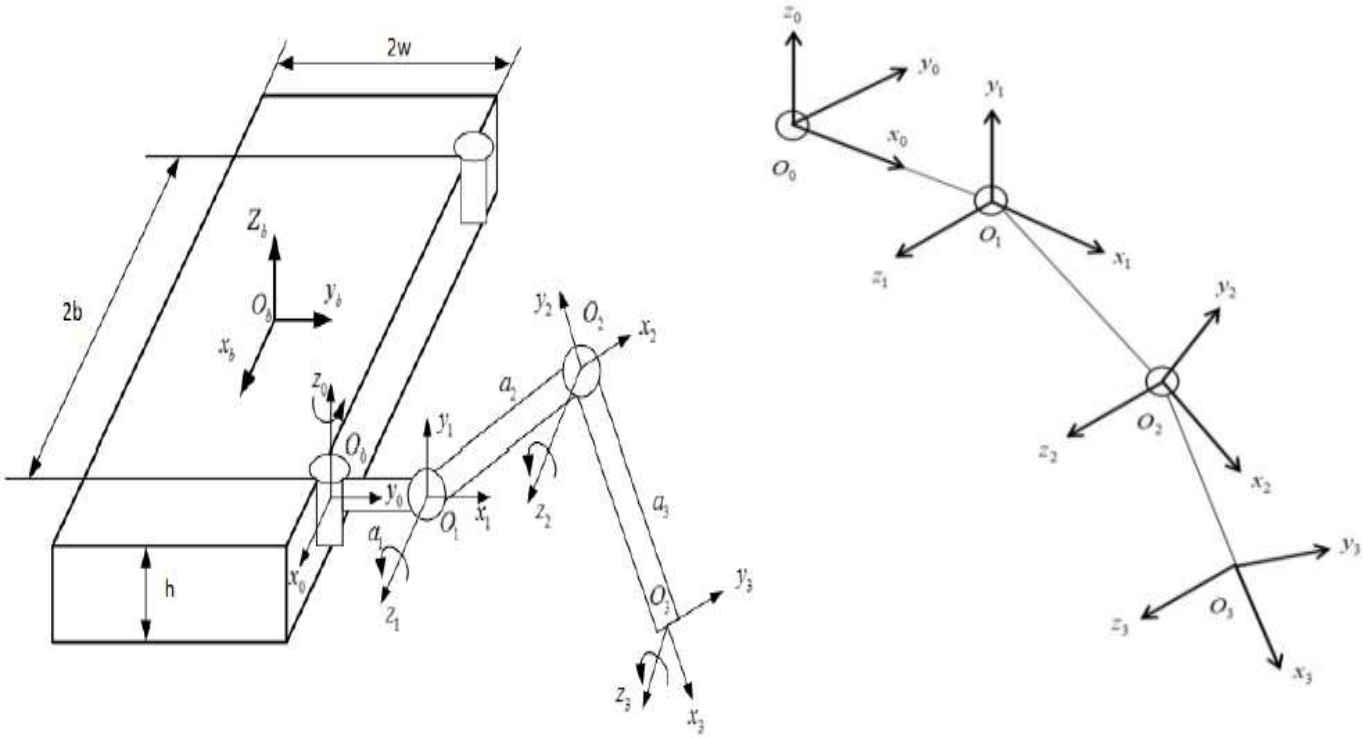


Figure 2. 5: The coordinate system diagram

After applied Denavit-Hartenberg (D-H) for kinematic analysis. Then inverse kinematics analysis refers to solving three joint rotation angles and obtaining inverse kinematics equations when the spatial position of the foot end of the mechanical leg is known. The inverse kinematic will transfer the new position into the rotation angle of servo . The root, femur and tibia are located in the same plane [3], so:

$$\theta_1 = \arctan(\frac{y_0}{x_0}) \qquad (2.\ 7)$$

From equation 2.6, we obtain:

$$\frac{x_0}{c_1} - l1 - l2.c_2 = l3.c_{23} \qquad (2.\ 8)$$

$$z_0 - l2.s_2 = l3.s_{23} \qquad (2.\ 9)$$

We add x_0, y_0, z_0 in equation 2.6 squared and substitute equation 2.7, equation 2.8 and equation 2.9. We obtain:

$$(2.x_0.l2.(\cos(\arctan(\frac{y_0}{x_0})))^{-1} - 2.l1.l2).c_2 + 2.z_0.l2.s_2 = x_0^2 + y_0^2 + z_0^2 - (l1)^2 - (l2)^2 - (l3)^2 - 2.x_0.l1(\cos(\arctan(\frac{y_0}{x_0})))^{-1} \qquad (2.\ 10)$$

The solution is:

$$\theta_2 = (x_0^2 + y_0^2 + z_0^2 - (l1)^2 + (l2)^2 - (l3)^2)\left((2.x_0.l2.(\cos(\arctan(\frac{y_0}{x_0})))^{-1} - 2.l1.l2\right)^2 + (2.z_0.l2)^2)^{-\frac{1}{2}}$$

$$-2.x_0.l1(\cos(\arctan(\frac{y_0}{x_0})))^{-1}((2.x_0.l2(\cos(\arctan(\frac{y_0}{x_0})))^{-1} - 2.l1.l2)^2 + (2.z_0.l2)^2)^{-\frac{1}{2}} \qquad (2.\ 11)$$

$$-\arctan(x_0.l2.z_0^{-1}l2^{-1}(\cos(\arctan(\frac{y_0}{x_0})))^{-1} - l1.l2.z_0^{-1}.(l2)^{-1})$$

We find θ_3,have:

$$\theta_3 = \arcsin(z_0.l3^{-1} - l2.l3^{-1}.s_2) - \theta_2 \qquad (2.\ 12)$$

2.3 THE METHOD OF CONTROLLING THE AXIS

Solving the positive and negative kinetic problem we were able to conduct motion programming for the robot legs, but to move in real environment, we need to study the movements of the robots, and the This requires the coordination of the 6-legged robot body.

With the rotation we have 3 directions in each of the x, y and z axes called roll, pitch and yaw respectively as shown in figure 2.6.

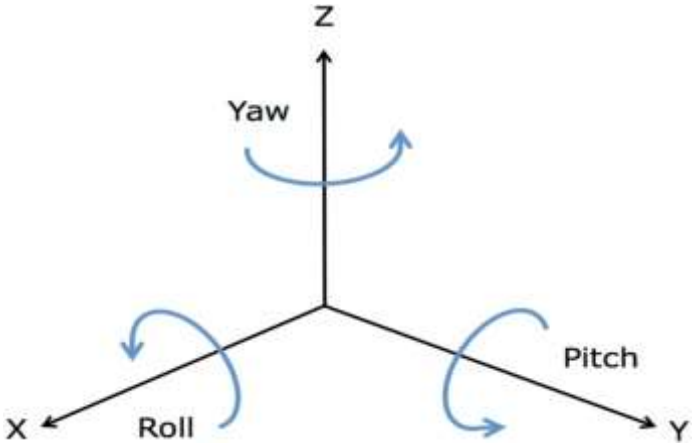


Figure 2. 6: Rotation motion in the axis.

To move according to Roll direction and Pitch direction is the same principle, is shown in figure 2.7 .

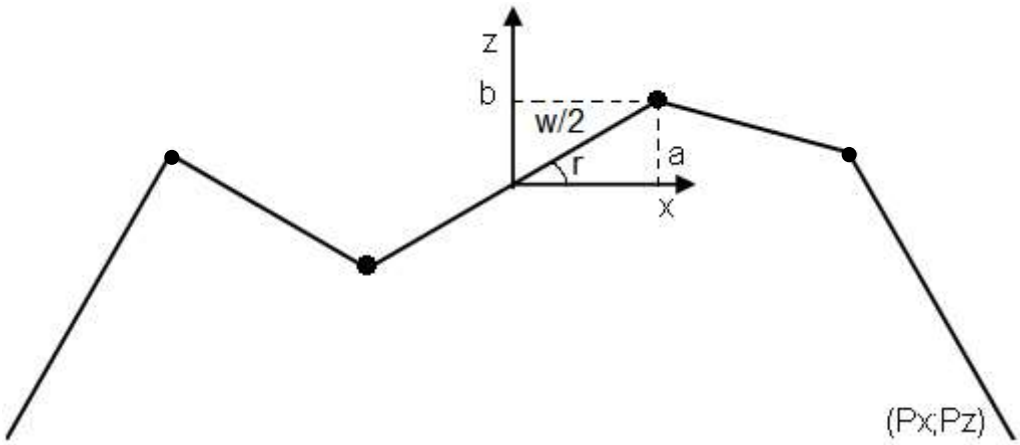


Figure 2. 7: The body rotation.

Using rotation matrix in 0xz coordinate[4]:

$$\begin{bmatrix} x' \\ z' \end{bmatrix} = \begin{bmatrix} \cos(r) & -\sin(r) \\ \sin(r) & \cos(r) \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} \quad (2.\ 13)$$

We obtain a result as:

$$\begin{cases} a = x' = x.\cos(r) - z.\sin(r) \\ b = z' = x.\sin(r) + z.\cos(r) \end{cases} \quad (2.\ 14)$$

$$\begin{cases} Px' = Px + (R - a) \\ Pz' = Pz + b \end{cases} \quad (2.\ 15)$$

Similarly with the remaining pins, when the 6 legs are updated to the new position, it will create a rotating motion.

When the robot rotates in the yaw angle, with the yaw angle, all 6 legs will simultaneously multiply with a rotation matrix along the Z axis of the middle body and update the new position to create a rotating motion. The new position is calculate by equation 2.16.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(yaw) & -\sin(yaw) \\ \sin(yaw) & \cos(yaw) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (2.\ 16)$$

We obtain a result as:

$$\begin{cases} x' = x.\cos(yaw) - y.\sin(yaw) \\ y' = x.\sin(yaw) + y.\cos(yaw) \end{cases} \quad (2.\ 17)$$

$$\begin{cases} Px' = x' \\ Py' = y' \end{cases} \quad (2.\ 18)$$

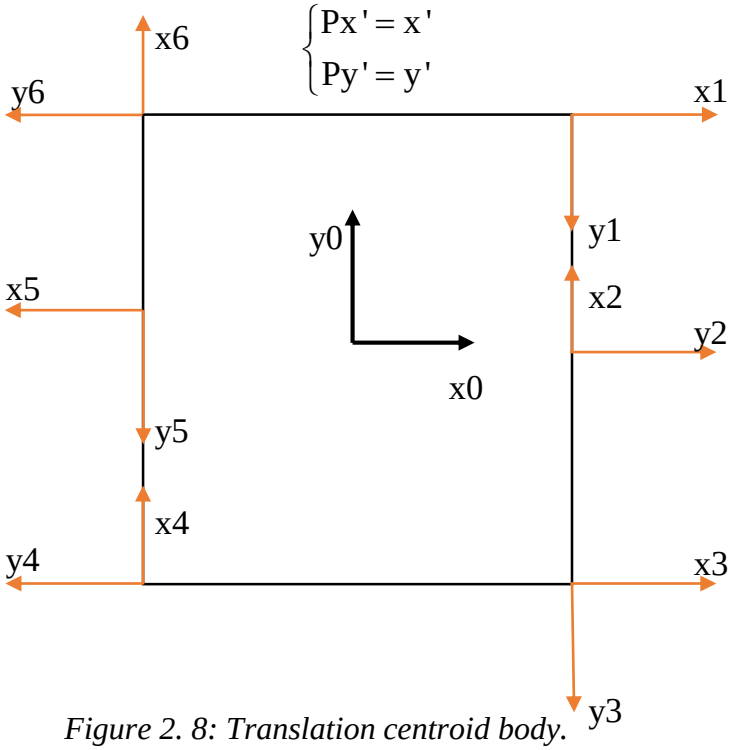


Figure 2. 8: Translation centroid body.

To translate centroid according y0 direction with a m-distance:

We calculated a new position:

$$\begin{cases} P'_{y1} = P_{y1} + m \\ P'_{x2} = P_{x2} - m \\ P'_{y3} = P_{y3} + m \\ P'_{x4} = P_{x4} - m \\ P'_{y5} = P_{y5} + m \\ P'_{x6} = P_{X6} - m \end{cases} \quad (2.\ 19)$$

To translate centroid according x0 direction with m-distance:

We obtain:

$$\begin{cases} P'_{x1} = P_{x1} + m \\ P'_{y2} = P_{y2} + m \\ P'_{x3} = P_{x3} + m \\ P'_{y4} = P_{y4} - m \\ P'_{x5} = P_{x5} - m \\ P'_{y6} = P_{y6} - m \end{cases} \quad (2.\ 20)$$

2.4 OPENCV LIBRARY

Figure 2. 9: The Logo's OpenCV.

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify

human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDAand OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a template interface that works seamlessly with STL containers.

In this project, We use the OpenCV library version 3.4.10 for Java programming on Android studio.

2.4.1 RGB TO HSV CONVERSATION

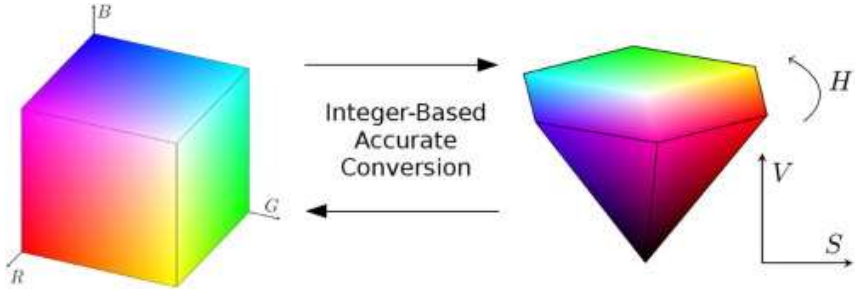


Figure 2. 10: RGB to HSV conversion

HSV components are often use as feature vectors for image retrieval and classification. HSV color space describes colors in terms of three components : Hue, Saturation and Value. Archana et al.applied HSV in multiple face detection method with accuracy of 93%. RGB to HSV conversion algorithm is required for video processing.

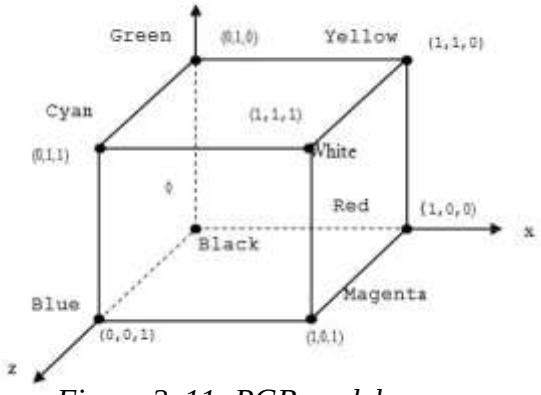


Figure 2. 11: RGB model.

The RGB color space is the most used color space for computer graphics. RGB color space describes colors in terms of the amount of red (R), green (G), and blue(B). RGB space can be visualized in the form of a cube in a three-dimensional Cartesian coordinate system in figure 2.11.

Each color-axis (R,G,B) is important component. The standard 24-bit RGB format is required for video processing. Total RGB pixel are 24-bit, the red, green, blue use 8-bit each. The total number of possible colors is 2^8*2^8*2^8 = 16.777.216. The components are unsigned integers in range [0,255]. However, in this conversion algorithm, RGB channels are required to be floating-point number in range [0,1] as in image figure 2.11.

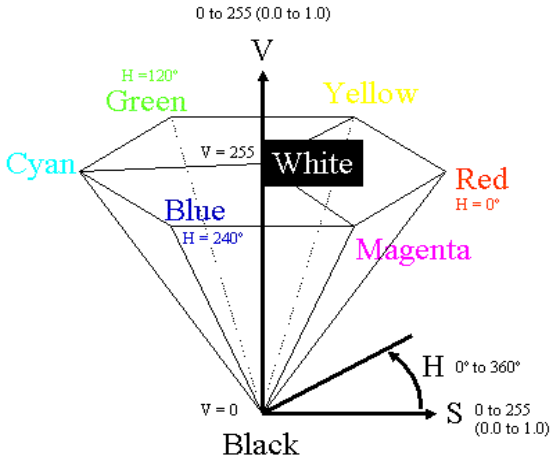


Figure 2. 12: HSV model.

The HSV color space in terms of three components: hue (H), saturation (S), and value (V). HSV models describes colors similarly to the human color sensation. The hue components (H) presents a true color (red, yellow, green, cyan, blue, magenta,).

The saturation component (S) represents color purity. It gives a measure of how much the true color is diluted by white. Decreasing saturation (S) corresponds to increasing whiteness.

The value component (V) is value of brightness. It measure the departure of the true color from black. Decreasing value (V) corresponds to increasing blackness. The HSV is different coordinate system describing the RGB model. The HSV is represent as a hexagonal cone.

The horizontal cross-section of hexaone have many different sizes. The size of the hexagons depends on value of V (value).According to the classical layout, 0^0 is red, 60^0 is yellow, 120^0 is green, 180^0 is cyan, 240^0 is blue, and 300^o is magenta.

This time , RGB to HSV conversion is using HSV hexaone. The algorithm use for RGB to HSV conversion in paper[]. The maximum and minimum are chosen from R,G,B which were inputted for the conversation. Calculate color space for HSV is seen in equation 2.21 to 2.27 [5]:

$$R' = \frac{R}{255} \; ; \; G' = \frac{G}{255} \; ; \; B' = \frac{B}{255} \tag{2. 21}$$

$$C_{max} = MAX(R', G', B') \tag{2. 22}$$

$$C_{min} = MIN(R', G', B') \tag{2. 23}$$

$$\Delta = C_{max} - C_{min} \tag{2. 24}$$

$$H = \begin{cases} 60^o \cdot \left(\frac{G' - B'}{\Delta} \bmod 6 \right), \; C_{max} = R' \\ 60^o \cdot \left(\frac{B' - R'}{\Delta} + 2 \right), \; C_{max} = G' \\ 60^o \cdot \left(\frac{R' - G'}{\Delta} + 4 \right), \; C_{max} = B' \end{cases} \tag{2. 25}$$

$$S = \begin{cases} 0, \; \Delta = 0 \\ \frac{\Delta}{C_{max}}, \; \Delta <> 0 \end{cases} \tag{2. 26}$$

$$V = C_{max} \tag{2. 27}$$

2.4.2 THRESHOLD MASK

In order to detect isolated color from the HSV image, We must to apply multiple mask. The pixel are partitioned depending upon their intensity value. We have a low threshold and high threshold mask for hue, saturation and value. Thresholding is the process of converting an image to binary image. If a pixel passes the threshold, it returns white (255), else returns black (0). The threshold algorithm is defined mathematical in equation 2.28 and equation 2.29[6].

$$g(x,y) = \begin{cases} 1 \text{ if } \; f(x,y) > \text{Threshold} \\ 0 \text{ if } \; f(x,y) < \text{ Threshold} \end{cases} \tag{2. 28}$$

Multiple thresholding divides pixels into multiple partitions with more Threshold values as shown in …

$$g(x,y) = \begin{cases} a, \text{ if } f(x,y) > \text{Threhold2} \\ b, \text{ if } \text{Threshold1} < f(x,y) \le \text{Threhold2} \\ c, \text{if } \; f(x,y) \le \text{Threshold1} \end{cases} \tag{2. 29}$$

Figure 2. 13: The image of a ball.

The aim of applying threshold algorithm is separate out regions of an image corresponding to object which we want to analyze. This separation is based on the variation of intensity between the object pixels and the background pixels.

2.4.3 MORPHOLOGICAL TRANSFORMATIONS

Mathematical morphology is defined from two basic algorithm. They are Erosion and Dilation.

An essential part of the erosion and dilation operations is structuring element used to probe the input image. A structuring element is a matrix which have any arbitrary shape and size.

2.4.4 KERNEL STRUCTURING ELEMENT - KERNEL

It is two-dimensional matrix, structuring element are typically much smaller than the image being processed. The center of structuring element is called origin. Matrix is binary matrix. Shapes of structuring element are rectangular, elliptical, cross-shaped. The rectangular is used for processing input image from camera cellphone.

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Figure 2. 14: The rectangular kernel.

2.4.5 EROSION

Erosion is the basic operation of the mathematical morphology. It is applied for binary image. The effect of operator is to erode away boundaries of regions of foreground pixels (pixel 1). The size of foreground pixels is shrined, and holes within those area become larger.

The erosion operator uses two input data. The first is original image (binary matrix). The second is structuring element – kernel matrix. To erose image, we convolve the binary image (original image) with a binary structuring element and then select binary matrix output.

Suppose that A is corresponding to the input binary image, and that B is structuring element. The erosion of A and B can be written as [7]

$$A \ominus B = \left\{ z \,|\, (B)_z \subseteq A \right\} \qquad (2.\ 30)$$

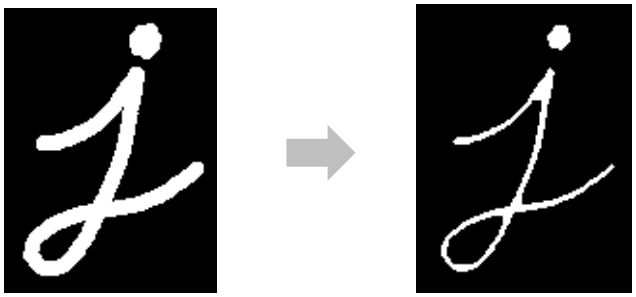


Figure 2. 15: After applied erosing for image.

2.4.6 DILATION

Dilation is the basic operation of the mathematical morphology. It applied for binary image. The basic effect of the operator is to gradually enlarge the boundaries of the regions of foreground pixels. The area of foreground pixels are increased and holes within those area become smaller.

The dilation operator uses two input data. The first is original image (binary matrix). The second is structuring element – kernel matrix. To dilate image, , we convolve the binary image (original image) with a binary structuring element and then select binary matrix output. Suppose that A is corresponding to the input binary image, and that B is structuring element. The erosion of A and B can be written as[7]:

$$A \oplus B = \left\{ z \,|\, \left[\left(\hat{B} \right)_z \cap A \right] \subseteq A \right\} \qquad (2.\ 31)$$

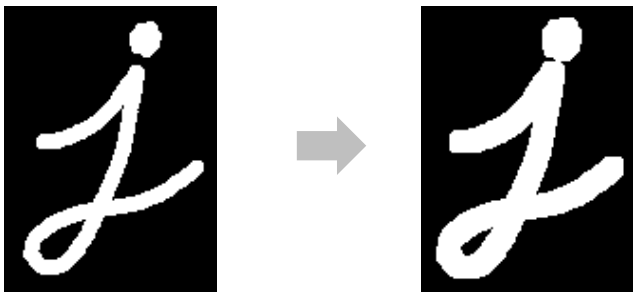


Figure 2. 16: After dilating for image.

2.4.7 CLOOSING IMAGE

Closing is similar in some ways to dilation in that it tends to enlarge the boundaries of foreground regions in an image, and shrink background color holes in such region, but it is less destructive of the original boundary shape.

-Take a structuring element: matrix kernel(3x3)

0	**1**	**0**
1	**1**	**1**
0	**1**	**0**

-To close an original matrix.

0	0	0	0	0	0	0	0
0	0	0	0	**1**	**1**	**1**	0
0	0	0	0	**1**	**1**	**1**	0
0	0	0	0	0	0	0	0
0	**1**	**1**	**1**	0	0	0	0
0	**1**	**1**	**1**	0	0	0	0
0	**1**	**1**	**1**	0	0	0	0
0	0	0	0	0	0	0	0

-After applied dilation algorithm.

0	0	0	0	1	1	1	0
0	0	0	1	**1**	**1**	**1**	1
0	0	0	1	**1**	**1**	**1**	1
0	1	1	1	1	1	1	0
1	**1**	**1**	**1**	1	0	0	0
1	**1**	**1**	**1**	1	0	0	0
1	**1**	**1**	**1**	1	0	0	0
0	1	1	1	0	0	0	0

Figure 2. 17: After closing for image.

-Final image after applied erosion algorithm.

0	0	0	0	1	1	1	0
0	0	0	0	**1**	**1**	**1**	0
0	0	0	0	**1**	**1**	**1**	0
0	0	0	1	1	0	0	0
0	**1**	**1**	**1**	0	0	0	0
0	**1**	**1**	**1**	0	0	0	0
0	**1**	**1**	**1**	0	0	0	0
0	0	0	0	0	0	0	0

2.4.8 OPENING IMAGE

Opening image is defined as an erosion followed by a dilation using the same structuring element.

-Take a structuring element : matrix kernel (3x3)

0	1	0
1	1	1
0	1	0

-To open an original image

0	0	0	1	1	1	0
0	0	0	1	1	1	0
0	0	0	1	1	1	0
0	0	1	1	0	0	0
1	1	1	0	0	0	0
1	1	1	0	0	0	0
1	1	1	0	0	0	0
0	0	0	0	0	0	0

Figure 2. 18: After opening for image.

-After applied srosion algorithm

0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

-Final image after applied dilation algorithm

0	0	0	0	0	1	0	0
0	0	0	0	1	1	1	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0

2.4.9 GAUSSIAN ALGORITHM

Gaussian blur is the application of the mathematical function to image in order to blur it. Gaussian blur the image to reduce the amount of noise, negligible details in an image, and remove speckles within the image. It is important remove the high frequency component.

Applying a Gaussian blur to an image is convolving an image with a Gaussian Kernel. This Gaussian in 2-D form is expressed as [8]:

$$G_{2D}(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (2.\ 32)$$

Where σ is the standard deviation of the Gaussian distribution, x is distance from origin in the horizontal axis, y is the distance from origin in the vertical axis. The value of σ controls the variance around a mean value of the Gaussian distribution, which determines the extent of the blurring effect around a pixel.

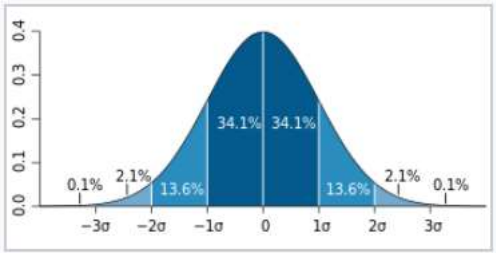


Figure 2. 19: Gaussian distribution function.

2.5 DETECTING CIRCLE USING HOUGH CIRCLE TRANFORM

2.5.1 CANNY EDGE DETECTION

Canny edge detection is a popular edge detection algorithm. The canny operator was designed to be an optimal edge detector. The canny operator includes 5 processes. The first is smoothing image by Gaussian convolution. Then calculating the gradient of image in order to detect the edge intensity and direction. The algorithm then remove the pixel value is non-maximum, a process known as non-maximum suppression. After application of non-maximum suppression, some pixels on the edge are noised. The image must be filtered out edge pixels with weak gradient value and preserve edge pixels with a high gradient value. The final steps is edge tracking by hysteresis, this algorithm has determined what strong edges and weak edges .The steps of canny edge detection algorithm in figure 2.20.

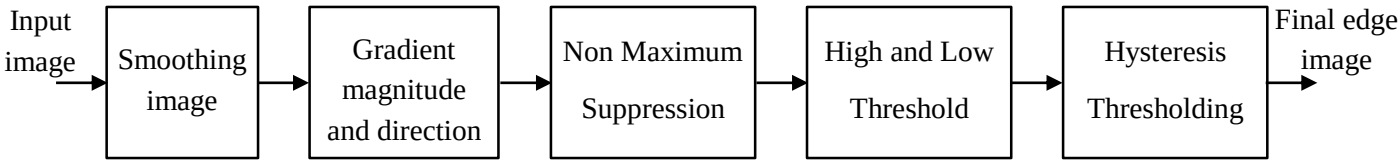


Figure 2. 20: Block diagram of Canny edge detection technique.

2.5.2 GAUSSIAN BLUR

Edge detection result will be affected by the image noise. One way to get rid of the noise on the image is applying Gaussian blur to smooth it. The Gaussian filter kernel is convolved with the image. Image convolution technique is applied with Gaussian kernel (3x3,5x5,9x9,….). A size (5x5) is a good size for most case. The equation for a Gaussian filter kernel of size (2k+1)x(2k+1) is given by[9]:

$$H_{ij} = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{\left(i-(k+1)\right)^2+\left(j-(k+1)\right)^2}{2\sigma^2}\right); 1 \le i,\ j \le (2k+1) \qquad (2.\ 33)$$

There is a 5x5 Gaussian filter with σ = 1.4 [10].

$$B = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \bullet A \qquad (2.\ 34)$$

2.5.3 INTENSITY GRADIENT

The Gradient calculation detects the edge intensity and direction by calculating the gradient of the image .To detect edge, the easiest way is to apply filters that highlight this intensity change in both directions – horizontal (x), and vertical (y).

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad (2.\ 35)$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad (2.\ 36)$$

Then, the magnitude G and the slope θ of the gradient are calculated as [11] :

$$G = \sqrt{G_x^2 + G_y^2} \qquad (2.\ 37)$$

$$\theta = \arctan\left(\frac{G_y}{G_x}\right) \qquad (2.\ 38)$$

Where G_x is derivative in the horizontal direction, G_y is derivative in the vertical direction. The edge direction angle is rounded to representing vertical, horizontal, and diagonals. The values of angle include 0, 45^o, 90^o and 135^o.

2.5.4 NON MAXIMUM SUPPRESSION

Non maximum-suppression is an edge thinning technique. The principle of operator is algorithm goes through all the points on the gradient intensity matrix and finds the pixels with the maximum value in the edge direction. The purpose of the algorithm is to check if the pixels on the same direction are more or less intense the ones being processed.

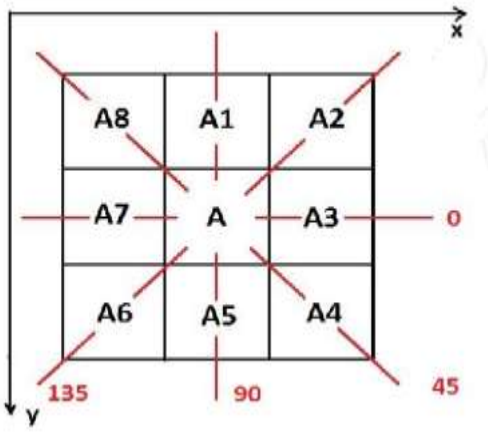


Figure 2. 21: Edge direction.

If the gradient angle is 0 degree($\theta = 0^o$), the edge direction is a horizontal line from left to right (A7-A-A3). In this example, the pixel (A) is being processed, and the pixels on the same direction are A7 and A3. If two pixels (A7; A3) are less intense than the pixel being processed. Then the intensity value of the current pixel is set to 1. Similarly, if the gradient angle is 90 degree. The edge direction is a vertical line from A1 to A5. Both pixels (A1, A5) are less intense than the pixel being processed. This pixel being processed (A) will be treat as an edge pixel. If the gradient angle is 135 degree, the edge direction is a diagonal line in the south west and north east direction. Both pixels (A2, A6) are less intense than the pixel being processed. The pixel A will be treat as an edge pixel[12].

2.5.5 DOUBLE THRESHOLD

The purpose of double threshold are identifying three kinds of pixels: strong, weak and non-relevant. Strong pixels are pixels that have an intensity high. Weak pixels are pixels that have an intensity that is not enough to evaluate. Other pixels are non-relevant edge.

- Upper threshold is used to identify the strong pixels.
- Lower threshold is used to identify the non-relevant pixels.
- All pixels have intensity between both threshold are weak pixels.

2.5.6 EDGE TRACKING BY HYSTERESIS

The effect of algorithm is transforming weak pixels into strong ones. The final image will have strong pixels. If the gradient magnitude of pixel is larger than upper threshold. A pixel will be treat as a strong edge pixel. If the gradient magnitude of the pixels is between both thresholds. That pixel will be treat as a weak edge pixel. Weak edges can be linked to the strong edges shown in figure 2.22.

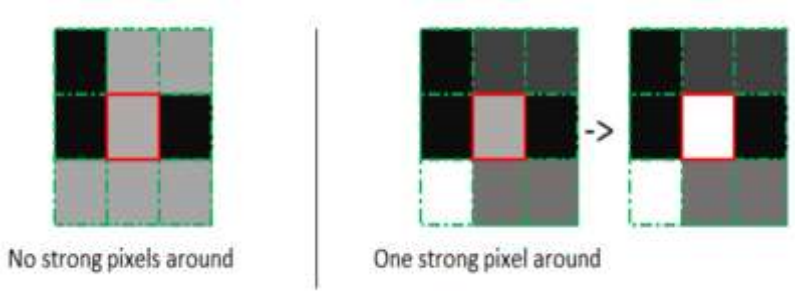


Figure 2. 22: The weak pixel link strong edge.

2.6 DETECT LOCATION OF CIRCLE

The last stage of image processing in this thesis is to detect the center and radius of the circles. The concepts of this stage relate to some concepts of Canny edge detection. Hence, this stage considers the input image which is a nose free grayscale image and don't have any random variation in intensity. There are three stage processes for detecting the location of the circle. The block diagram below will show how to obtain it:

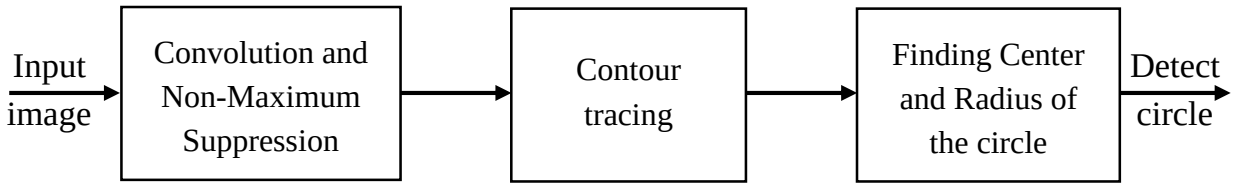


Figure 2. 23: The diagram of detection circle.

2.6.1 CONVOLUTION AND NON-MAXIMUM SUPPRESSION

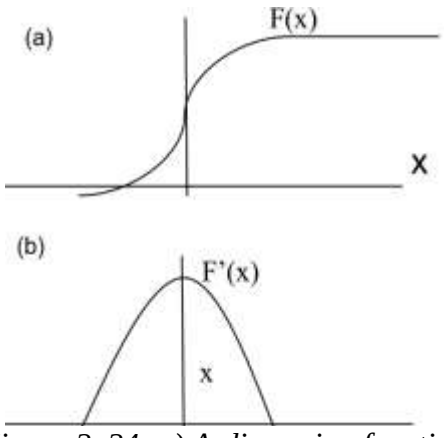


Figure 2. 24: a) A-dimension function in intensity; b) Derivative of function

The first stage is convolution and Non-Maximum Suppression. The input of this stage is grayscale image and the tasks of this stages are using horizontal and vertical kernels to obtain horizontal and vertical directions for each pixel. These values are rounded to one of the eight directions for each pixel. Besides that, these kernels are also used to get the edge strength for the image.

Based on the evidence that the edges in an image are expected to be the boundaries of objects that tend to produce sudden changes in the image intensity [13]. This method helps us to identify circles in an image more quickly with effective use of resources. For example, different objects in an image usually many different colors, hue or light intensity and this cause a change in image intensity. Thus, much of the geometric information that would be conveyed in a line drawing is captured by the intensity changes in an image. The input to this algorithm is assumed to be a noise free grayscale image. Hence, there is no very high random variation in intensity from one pixel to another.

The change in figure 2.24 shows how to detect edge by detecting the change in the intensity. The figure 2.24a is a 1-dimensional edge and after using the convolution kernel on this signal, we get figure 2.24b.

Figure 2.24b has the maximum located at the center of the edge of the original signal. Based on one dimensional analysis, the theory can be extended to two dimensions. For a grayscale digital image, four 3x1 window operators are used to get two pixel directions, Horizontal (Kx) and Vertical (Ky), for each pixel, taking into account the neighbors of the pixel:

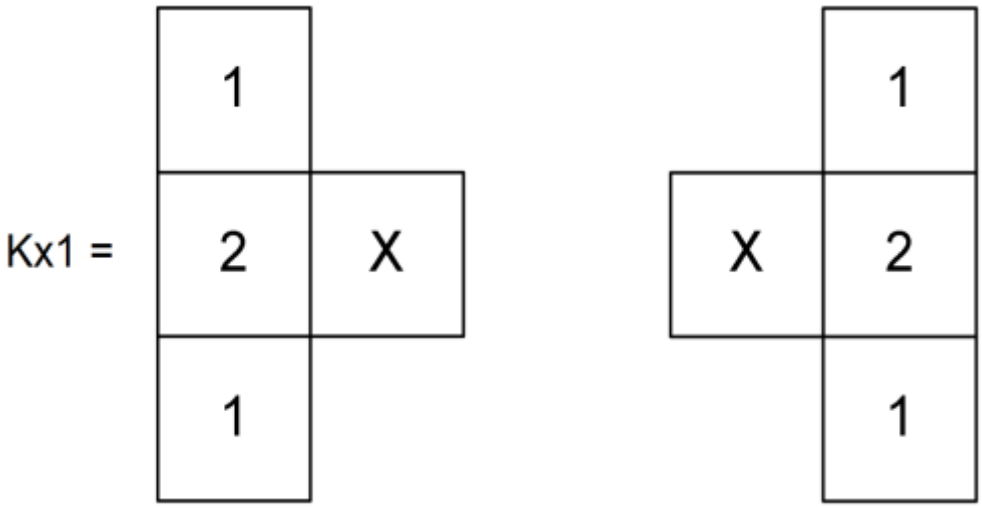


Figure 2. 25: The kernel for finding Horizontal gradient orientation.

X is the main factor of two kernels above [14].In Kx1 and Kx2, the value of the center of the first and the last column, are 2 while the rest values are 1. This is to emphasis on the

values of the horizontal neighboring pixels compared to the others. Similarly, below is kernels used to find vertical gradient orientation:

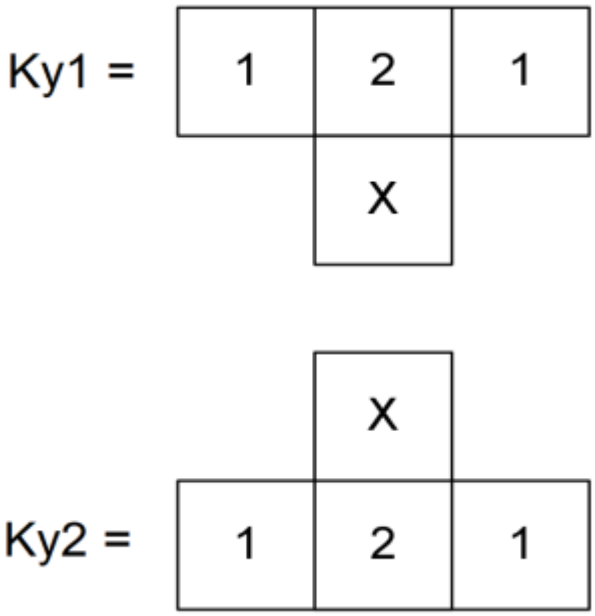


Figure 2. 26: Kernel for finding vertical gradient orientation.

X is the main factor of two kernels above. In Ky1 and Ky2, the values of the center of the first and the last row, are 2 while the rest values are 1. This is to emphasis on the values of the vertical neighboring pixels compared to the others. Kx1, Kx2, Ky1, Ky2 are used to get the horizontal and vertical pixel direction, pixel edge strength.

The next part, we will discuss how to calculate the pixel direction and edge strength for a pixel. To calculate pixel direction, we just need to calculate horizontal and vertical orientation. To calculate edge strength, horizontal and vertical orientation as well as horizontal and vertical gradient magnitude are calculated.

Equation (2.39) is the convolution of kernel Kx1 and Image I while equation (2.40) is the convolution of kernel Kx2 and Image I. These two sets of values, Ix1 and Ix2, determine the horizontal orientation of a pixel point (x, y) of the image I. Equation 2.41 show how to calculate horizontal gradient magnitude of the point I (x, y). Thus, these two pieces of need information, horizontal orientation and horizontal gradient magnitude, are obtained.

$$Ix1 = I.Kx1 \qquad (2.\ 39)$$

$$Ix2 = I.Kx2 \qquad (2.\ 40)$$

If Ix1(x, y) > Ix2(x, y) then the orientation I_h_ore is: ⟶

If Ix1(x, y) < Ix2(x, y) then the orientation I_h_ore is: ⟵

If Ix1(x, y) = Ix2(x, y) then the orientation is: 0

$$I\_hor(x,y) = \left|Ix2(x,y) - Ix1(x,y)\right| \qquad (2.\ 41)$$

Equation 2.43 is the convolution of kernel Ky1 and Image I while equation 2.44 is the convolution of kernel Ky2 and Image I. These two sets of values, Iy1 and Iy2, determine

the vertical orientation of a pixel point (x, y) of the image I. Equation 2.45 show how to calculate vertical gradient magnitude of the point I (x, y). Thus, these two pieces of need information, vertical orientation and vertical gradient magnitude, are obtained.

$$Iy1=I.Ky1 \quad (2.\ 43)$$

$$Iy2 = I.Ky2 \quad (2.\ 44)$$

If Iy1(x, y) > Iy2(x, y) then the orientation I_v_ore is: ↓

If Iy1(x, y) < Iy2(x, y) then the orientation I_v_ore is: ↑

If Iy1(x, y) = Iy2(x, y) then the orientation is: 0

$$I\_ver(x,y) = |Iy2(x,y) - Iy1(x,y)| \quad (2.\ 45)$$

The four needed pieces of information, horizontal orientation I_h_ore, vertical horizontal orientation I_v_ore, horizontal gradient magnitude I_hor(x, y), vertical gradient magnitude I_ver(x, y), are obtained for a pixel(x, y). By using horizontal and vertical gradient magnitude, the edge strength is calculated by the equation 2.46:

$$I\_mag = |I\_hor + I\_ver| \quad (2.\ 46)$$

The vertical and horizontal orientation for each point in the image is used to determine the pixel orientation. The pre-defined conditions used to get the pixel direction are as shown in Table 2.3:

Table 2. 3: Conditions used to determine the pixel direction.

No.	Vertical Orientation Iy	Horizontal Orientation Ix	Conditions	Range	Result
1	↑ Or ↓	→	I_hor>4*I_ver	346’-14’	1 →
2	↑	←	I_hor>4*I_ver I_ver<4*I_hor	14’-76’	2 ↗
3	↑	Or ← →	I_ver>4*I_hor	76’-104’	3 ↑
4	↑	←	I_hor<4*I_ver I_ver<4*I_hor	94’-166’	4 ↖
5	↑ Or ↓	←	I_hor>4*I_ver	166’-184’	5 ←
6	↓	←	I_hor<4*I_ver I_ver<4*I_hor	184’-256’	6 ↙
7	↓	Or ← →	I_ver>4*I_hor	256’-284’	7 ↓
8	↓	→	I_hor<4*I_ver I_ver<4*I_hor	284’-346’	8 ↘

All processes mentioned above are repeated for all the pixels in the image to get the pixel direction and edge strength. The pixel direction and edge strength are then used by Non-Maximum Suppression to determine thin edges for an object.

Non-Maximum Suppression is a set of thin edges point, in the form of a binary image, is obtained. After this process, edge width is only 1 pixel. The convolved image after obtained pixel direction and edge strength is scanned. When we find any pixel direction other than 0 and we look for other pixels in the same area having the same pixel direction. A linear window, as discussed above (in edge strength and pixel direction), having immediate neighbors of pixels under consideration, is used. Hence, the size of the linear window is 3 pixels. The pixels having the highest edge strength within the linear window is considered and the rest of the pixels in the linear window are set zero. Table 2.4 below is an example of using NMS:

Table 2. 4: Image with linear window in color corresponding to the pixel direction and gradient magnitude and result of NMS.

50	90	70	50	50
40	70	60	50	70
50	60	90	60	60
50	50	50	80	50
50	50	50	50	70

50	90	70	50	50
40	0	60	50	70
50	60	90	60	60
50	50	50	0	50
50	50	50	50	70

The linear table is displayed in red color. After scanning all pixel of I, we find that three directions of linear window are both 135 degrees. In NMS, we only retain the pixel which has the highest edge strength and the others of linear are set zero. Hence, in the linear window above, the pixel with edge strength 90 is retained and the edge strength and pixel direction of the rest of pixels are made zero. This provides us the thin edges of objects in an image. These thin edges found can form a boundary of a circular object in the image. The next process uses these thin edges to find the arcs which can be a part of a potential circle.

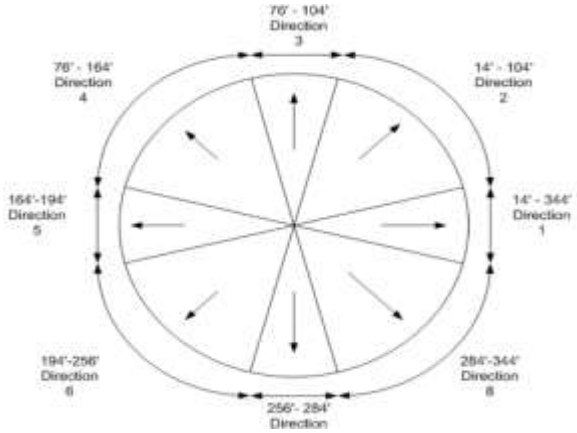


Figure 2. 27: Range of angles of pixel direction.

2.6.2 CIRCLE HOUGH TRANSFORM

The Circle Hough Transform (CHT) is a feature extraction technique for detecting circle. Detecting circles in an image are one of the problems that are discussed in this paper. Many algorithms, such as Linear Square Method (Hsiao et al., 2006), Hough Transform, and Canny Edge detection Algorithms have been proposed to detect circles. These algorithms detect circles from the edge detected images. Among these algorithms, Early Circular Hough Transform has been widely successful in meeting the real-time requirement of being able to detect the circles in noisy environments.

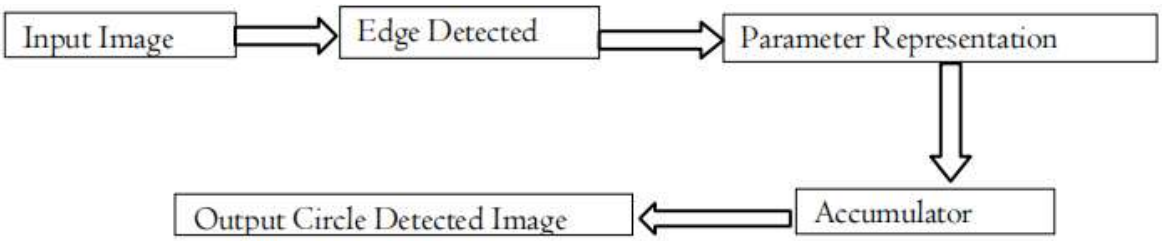


Figure 2. 28: Circular Hough Transform Algorithm.

A circle with center (a,b) and radius r, in a binary image, is specified by the parameters (a,b,r) in the equation[15]:

$$(x-a)^2 + (y-b)^2 = r^2 \qquad (2.\ 47)$$

with (x,y) the set edge pixels that make up the circumference of this circle. The parametric representation of the circle is:

$$\begin{cases} x = a + r\ \cos\theta \\ y = b + r\ \sin\theta \end{cases} \qquad (2.\ 48)$$

The circle has three parameter r, a and b, hence the parameter space will belong to R^3 . When we applied traditional CHT technique to industrial images which is obtained from Palletization plant, the results obtained contains many false circles which may be due to pellets which looks similar to circle but not exactly the circle. For such type of real industrial images we extended the concept of local maxima to different accumulators.

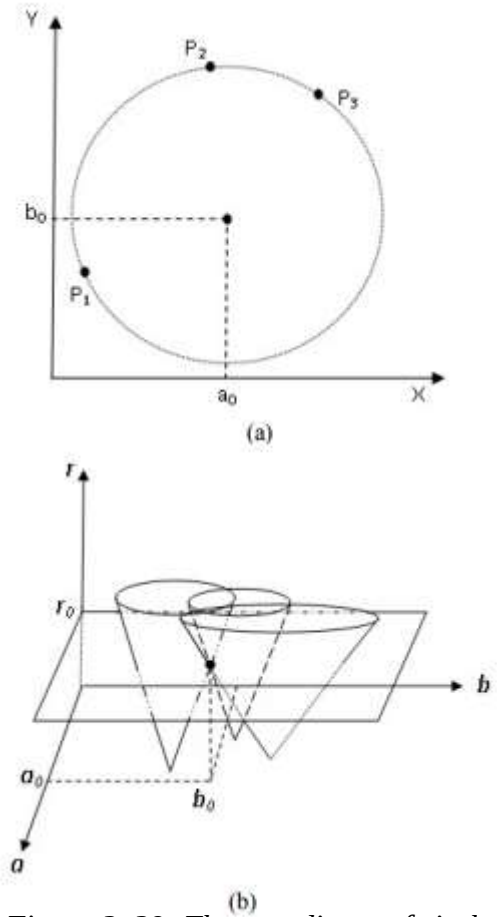


Figure 2. 29: The coordinate of circle.

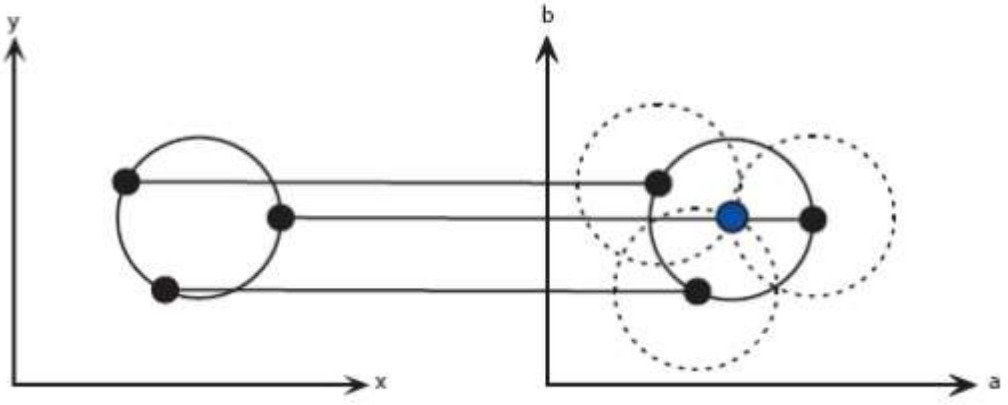


Figure 2. 30: Parametric space representation of a constant radius circle.

First we find all edges using any of the edge detection technique in the image. We have used canny edge detection technique. The radius range is either given by the user of the application or already defined in program. Suppose radius range defined in program is (R_i, R_j). Then R_j - R_i will give the number of accumulators to be made and searched out in order to carry out desired task of detection. Accumulator matrix will have the same size as the parameter space.

Now consider each edge pixel obtained as result after applying canny edge detection technique. For each edge pixel draw a circle having radius equal to R_i (R_i lie within the radius range).The circle is drawn in parameter space. Here avalue corresponds to x-axis and b-value corresponds to y-axis while the z-axis is the r. For this radius R_i we have one distinct accumulator. Initially all the cell values in the accumulator are set to zero. At the coordinates which lie on the perimeter of the drawn circle in parametric space, increment the value in accumulator matrix. Sweep every edge point in the input image and increment the cell values in the accumulator matrix.

This process is repeated for all desired r which belong to radius range. If radius range is (R_i, R_j) then total number of accumulators needed is R_j - R_i.

A_1=

64	2	3	61	60	6	7	57
9	55	54	12	13	51	50	16
17	47	46	20	21	43	42	24
40	26	27	37	36	30	31	33
32	34	35	29	28	38	39	25
41	23	22	44	45	19	18	48
49	15	14	52	53	11	10	56
8	58	59	5	4	62	63	1

Figure 2. 31: Accumulator matrix sample.

Each accumulator which is made for particular radius will contain numbers also called votes or cuts. The number in any cell of accumulator matrix indicates the number of circles passing through that individual coordinates. First apply the concept of local maxima within the same accumulator. For each row of accumulator matrix, find out the local maxima. Figure shown above contain one sample of accumulator matrix of order 8 x 8. Consider first row of $A_{1.}[15]$.

$$A_l\,(1, 1{:}8) = \{64, 2, 3, 61, 60, 6, 7, 57\}$$

First row of Al contains 8 elements. Taking window size equals to five, slide this window to entire row over each element to find the local maxima .

Figure 2. 32: Sliding window on first row of AI.

From the above, only one maxima found at the fist row and fourth column. Corresponding coordinates (1, 4) is temporary elected as center candidate. This elected candidate has to pass through one addition tests to confirm the election for center candidate.

Keep going with another accumulator, sweep over the energy edge point in the input image drawing circle with the desired circle with desired r and incrementing the value in our accumulator. When every edge point and every desired radius is used we can turn our attention to accumulator ill now contain numbers corresponding to the number of circles passing through the individual coordinate. Thus the highest number corresponds to the circle of the circle in the image.

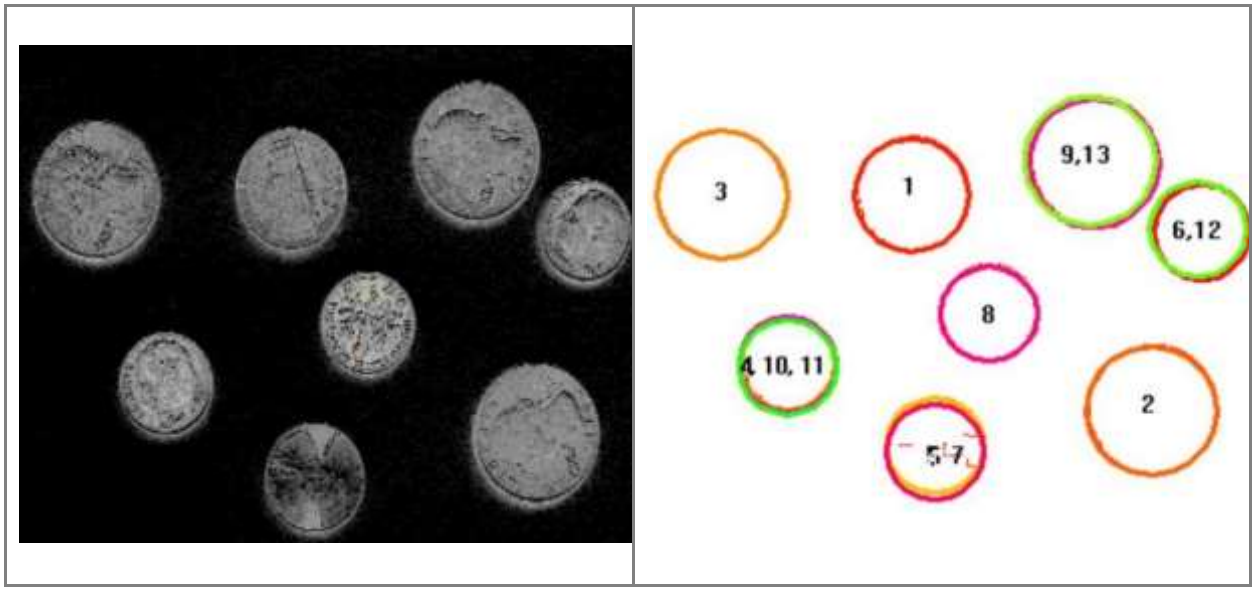


Figure 2. 33: Detect circle of the coins.

2.6.3 FINDING CENTER AND RADIUS OF THE CIRCLE

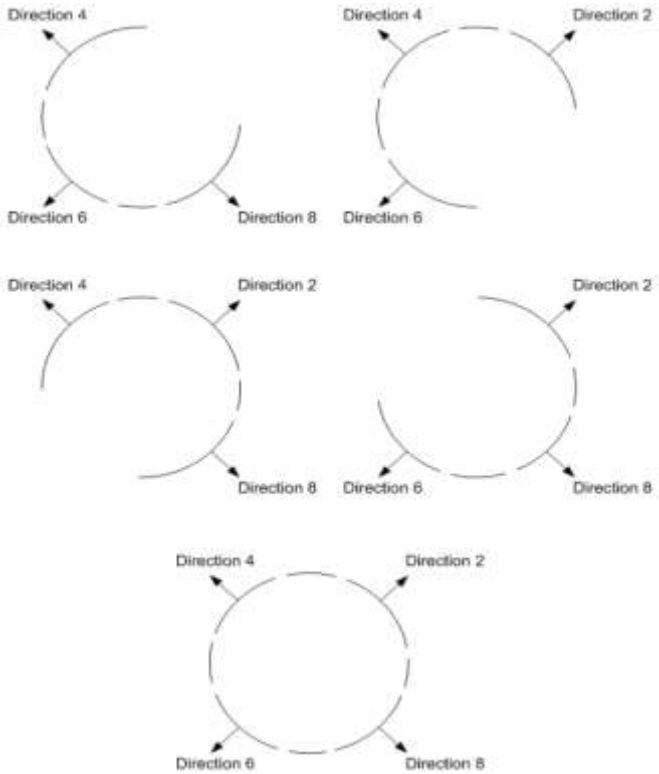


Figure 2. 34: All the combinations of major arcs which can form a circle.

First, we need to find the center of circle by using the property of the circle which state, “Perpendicular bisector of a chord contains the center of the circle” [16].

From the arcs stored, depending on the position of the arc, either arc with direction 2 and arc with direction 8 (or arc with direction 6 and arc with direction 4) are scanned to find the points B and C, points with same x value in the spatial domain. Midpoint of BC

give us the y0 value of the center, as mentioned by the property of the circle. This procedure is carried out for all the points in the arcs to find an array of y0 values for the center of the circle. Similarly, depending on the position of the arc, either arc with direction 2 and arc with direction 4 (or arc with direction 8 and arc with direction 6) are scanned to find the points A and B, points with same y value in the spatial domain. Midpoint of AB give us the x0 value of the center, as mentioned by the property of the circle. This procedure is carried out for all the points in the arcs to find an array of x0 values for the center of the circle. Hence, we have the array of the x0 and y0 values for the center of the circle for the given values of arcs. The value of x0 and y0 which are repeated most number of times are noted. If these x0 and y0 values have been repeated more than 50% of the length of the array, then the value is considered as the center of the circle. Figure 2.35 illustrates how to find the center of the circle using the perpendicular bisector property of the chord.

Next, to find the radius of the circle, the distance between the arc points and the points on the circumference of the circle give us the radius of the circle. If x0 is the x value of the center of the circle and x1 is the x value on the arc with direction 1 or 5, then the radius r2 is the difference between x1 and x0. And if y0 is the y value of the center of the circle and y1 is the y value on the arc with direction 3 or 7, then the radius r1 is the difference between y1 and y0. If the both values of radius found are the same, then it is confirmed that these arcs form a circle. Figure 2.36 shows us how to find the radius by using with pixel direction 3 and pixel direction 1.

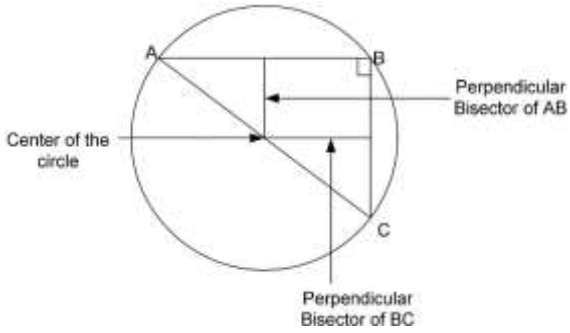


Figure 2. 35: Finding center of the circle using the perpendicular bisector property of the chord.

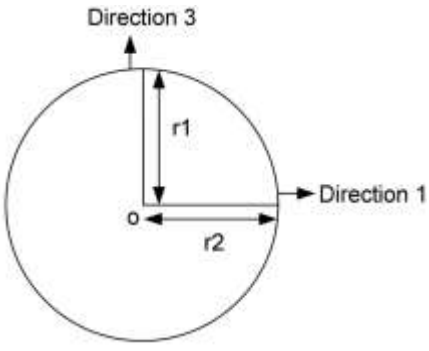


Figure 2. 36: R1 and R2 radius found using the arc with pixel direction 3 and 1 respectively.

2.7 DATA TRANSMISSION

Wireless communication technology is one of the most popular communications. Mobile technology, robot and other devices can be connected together by wireless communication technology. In this thesis, we will use Bluetooth technology and the Firebase technology. The Smartphone is used to control the robot movement and monitor the surrounding environment. Wireless technology provide many benefits : faster response, real time control, tighter communication between client and host.

2.7.1 FIREBASE TECHNOLOGY

Firebase is a powerful platform for your mobile and web application. Firebase can power your app's back-end, including data storage, user authentication, static hosting, and more. With Firebase, you can easily build mobile and web apps that scale from one user to one million.

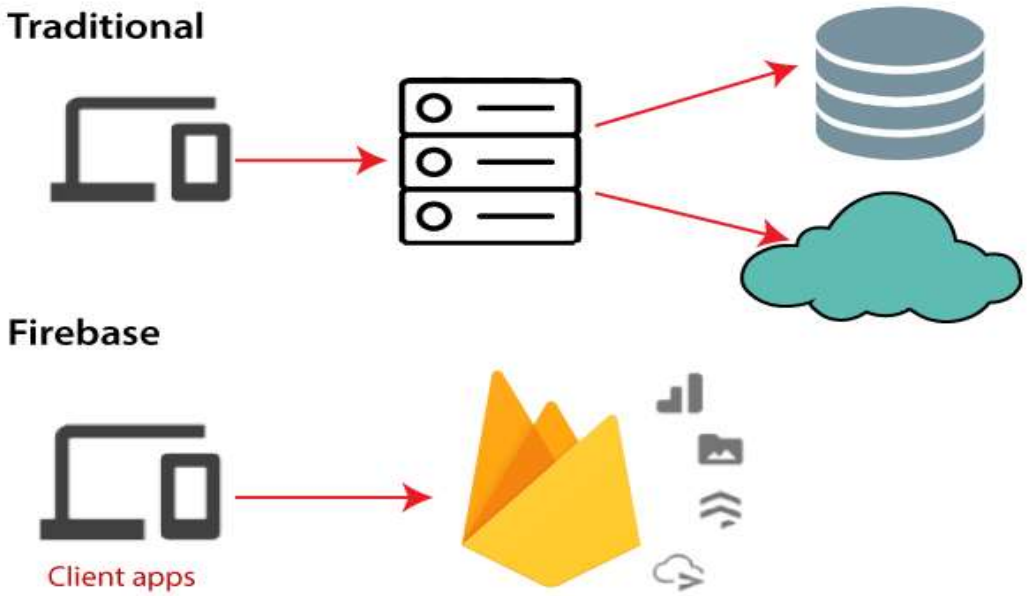


Figure 2. 37: The communication of Firebase.

Back in 2011, before **Firebase** was Firebase, it was a startup called Envolve. As Envolve, it provided developers with an API that enabled the integration of online chat functionality into their website.

What's interesting is that people used Envolve to pass application data that was more than just chat messages. Developers were using Envolve to sync application data such as a game state in real time across their users.

This led the founders of Envolve, James Tamplin and Andrew Lee, to separate the chat system and the real-time architecture. In April 2012, Firebase was created as a separate company that provided Backend-as-a-Service with real-time functionality.

After it was acquired by Google in 2014, **Firebase** rapidly evolved into the multifunctional behemoth of a mobile and web platform that it is today.

Figure 2. 38: The features in Firebase.

Firebase helps mobile appteams succeed. With firebase you can:

- Build better apps.
- Improve app quality.
- Grow your business.

2.7.1.1 BUILD BETTER APPS

Firebase lets you build more powerful, secure and scalable apps, using world-class infrastructure using:

1. Cloud Firestore: is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud Platform. It is a NoSQL document database that lets you easily store, sync, and query data for your mobile and web apps — at a global scale.It's supporting for Android, iOS and Web Platform.

2. ML Kit: is a mobile SDK that brings Google's machine learning expertise to Android and iOS apps in a powerful yet easy-to-use package. Whether you're new or experienced in machine learning, you can implement the functionality you need in just a few lines of code. There's no need to have deep knowledge of neural networks or model optimization to get started. On the other hand, if you are an experienced ML developer, ML Kit provides convenient APIs that help you use your custom TensorFlow Lite models in your mobile apps. It's supporting for Android and iOS Platform.

3. Cloud Functions: for Firebase lets you automatically run backend code in response to events triggered by Firebase features and HTTPS requests. Your code is stored in Google's cloud and runs in a managed environment. There's no need to manage and scale your own servers. It's supporting for Android, iOS, C++, Unity and Web Platform.

4. Authentication: provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more. It's supporting for Android, iOS and Web Platform.

5. Hosting: is production-grade web content hosting for developers. With a single command, you can quickly deploy web apps and serve both static and dynamic content to a global CDN (content delivery network). You can also pair Firebase Hosting with Cloud Functions to build and host micro services on Firebase. It's supporting only Web Platform.

6. Cloud Storages: is for object storage service built for Google scale. The Firebase SDKs for Cloud Storage add Google security to file uploads and downloads for your Firebase apps, regardless of network quality. You can use our SDKs to store images, audio, video, or other user-generated content. On the server, you can use Google Cloud Storage, to access the same files. It's supporting for Android, iOS, C++, Unity and Web Platform.

7. Real-time Database: is a cloud-hosted NoSQL database that lets you store and sync between your users in real-time. The Real-time Database is really just one big JSON object that the developers can manage in real-time. It's supporting for Android, iOS, C++, Unity and Web Platform.

Figure 2. 39: The logo's Firebase.

The Firebase Real-time Database is a cloud-hosted NoSQL database that lets you store and sync between your users in real-time.

The Real-time Database is really just one big JSON object that the developers can manage in real-time.

Figure 2. 40: Real-time Database be become A Tree of Values.

With just a single API, the Firebase database provides your app with both the current value of the data and any updates to that data.

Real-time syncing makes it easy for your users to access their data from any device, be it web or mobile. Real-time Database also helps your users collaborate with one another.

Another amazing benefit of Real-time Database is that it ships with mobile and web SDKs, allowing you to build your apps without the need for servers.

When your users go offline, the Real-time Database SDKs use local cache on the device to serve and store changes. When the device comes online, the local data is automatically synchronized.

The Real-time Database can also integrate with Firebase Authentication to provide a simple and intuitive authentication process.

2.7.1.2 AUTHENTICATION

Figure 2. 41: The Firebase application connect the other applications on smartphone.

Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app.

Normally, it would take you months to set up your own authentication system. And even after that, you would need to keep a dedicated team to maintain that system. But if you use **Firebase**, you can set up the entire system in under 10 lines of code that will handle everything for you, including complex operations like account merging.

You can authenticate your app's users through the following methods:

- Email & Password
- Phone numbers
- Google
- Facebook
- Twitter

Using **Firebase Authentication** makes building secure authentication systems easier, while also improving the sign-in and onboarding experience for end users.

Firebase Authentication is built by the same people who created Google Sign-in, Smart Lock, and Chrome Password Manager.

2.7.2 FIREBASE CONFIGURATION

An API key or application programming interface key is a code that gets passed in by computer applications. The program or application then calls the API or application programming interface to identify its user, developer or calling program to a website.

Application programming keys are normally used to assist in tracking and controlling how the interface is being utilized. Often, it does this to prevent abuse or malicious use of the API in question. An API key can act as a secret authentication token as well as a unique identifier. Typically, the key will come with a set of access rights for the API that it is associated with.

To access and transmit information between devices wherever having internet, Firebase is used as a cloud storage allows devices to connect to a web API, which can be controlled from a far away distance in real time, here are the steps to establish our Hexapod app connection with Google Firebase:

- Step 1: Create a project:

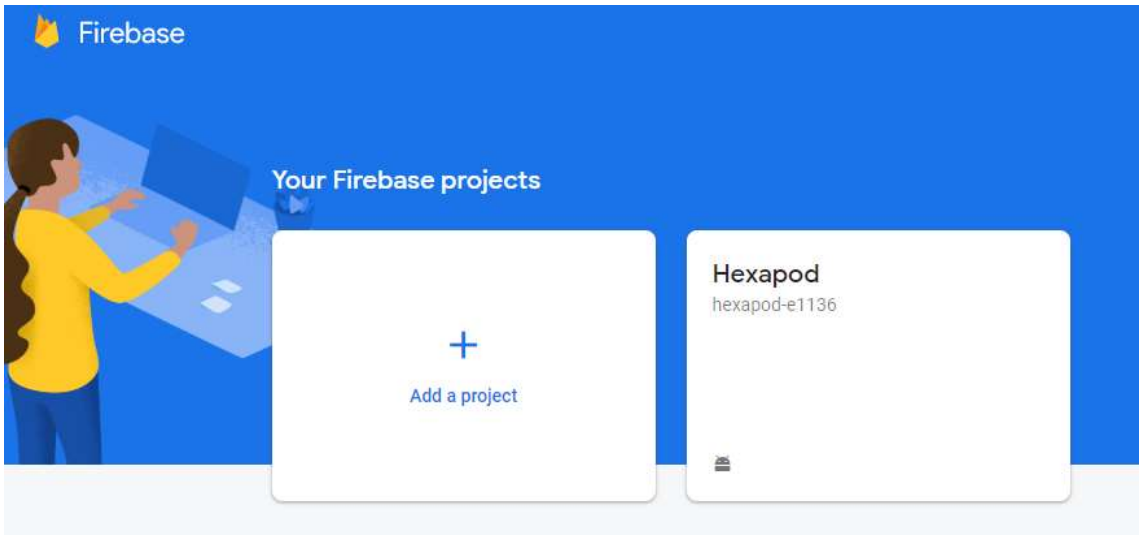


Figure 2. 42: Create a project Hexapod.

- Step 2: Selecting the database option ,scroll down in the next page and select create real-time database. And create it in the locked mode.

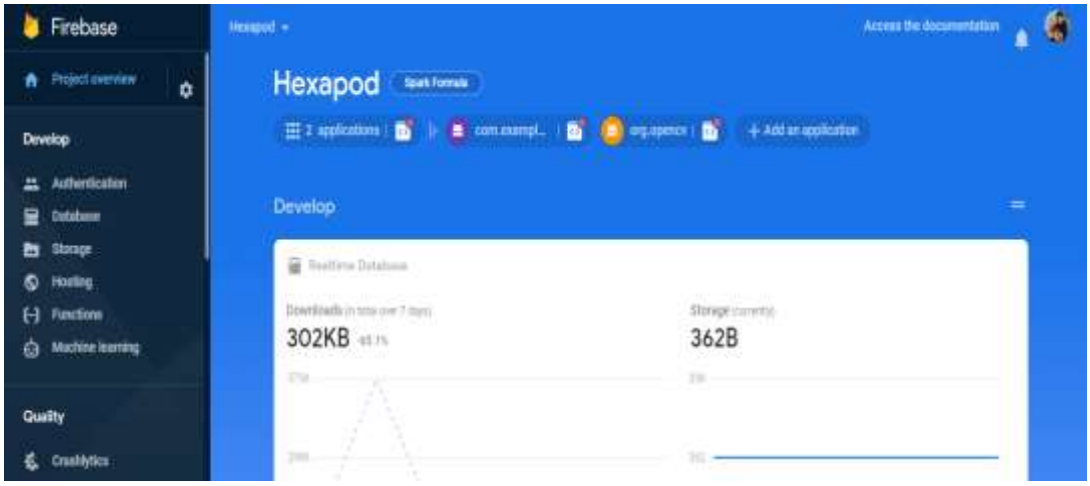


Figure 2. 43: The Real-time database.

- Step 3: Then we need to add fields to this database. Unlike normal relational databases like MySQL , Oracle , Firebase is an NoSQL database. In here data is stored in JSON format . So it is very easy for applications to access data stored in this.

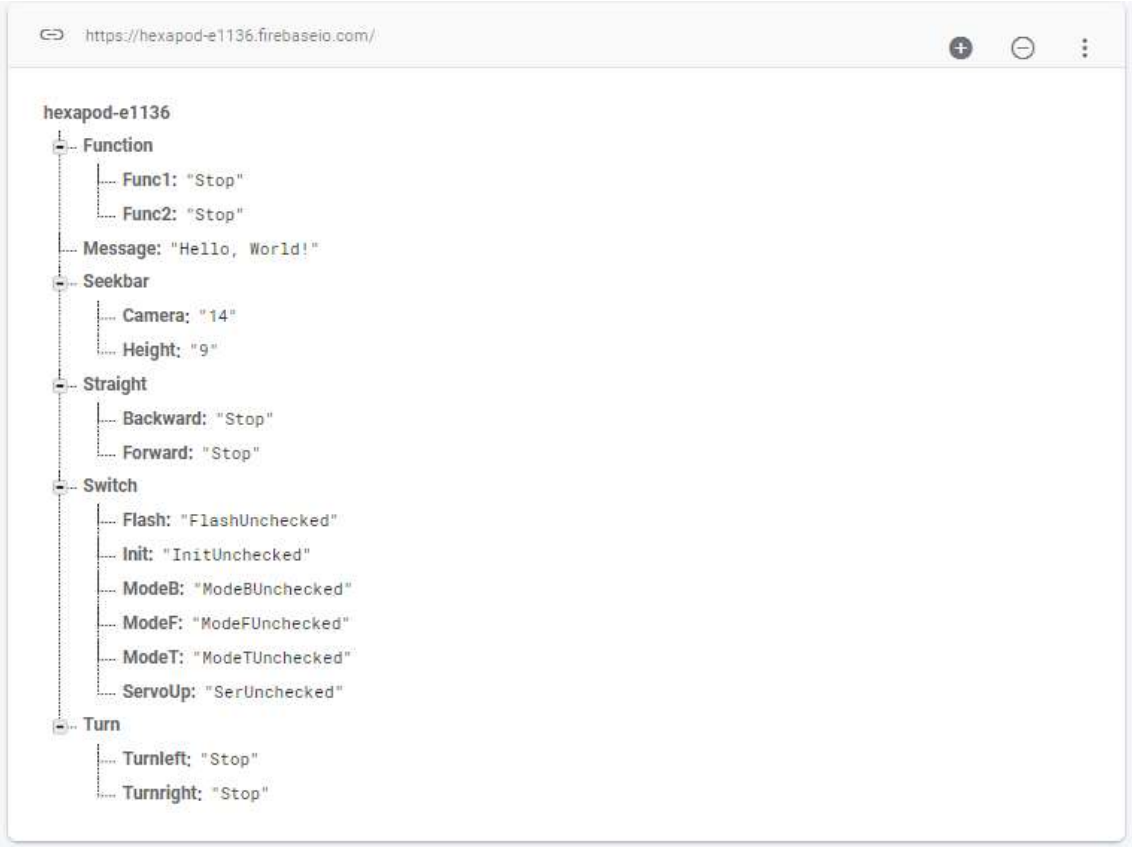


Figure 2. 44: The manage structure data Firebase.

- Step 4: Add configuration file to Android Studio application.

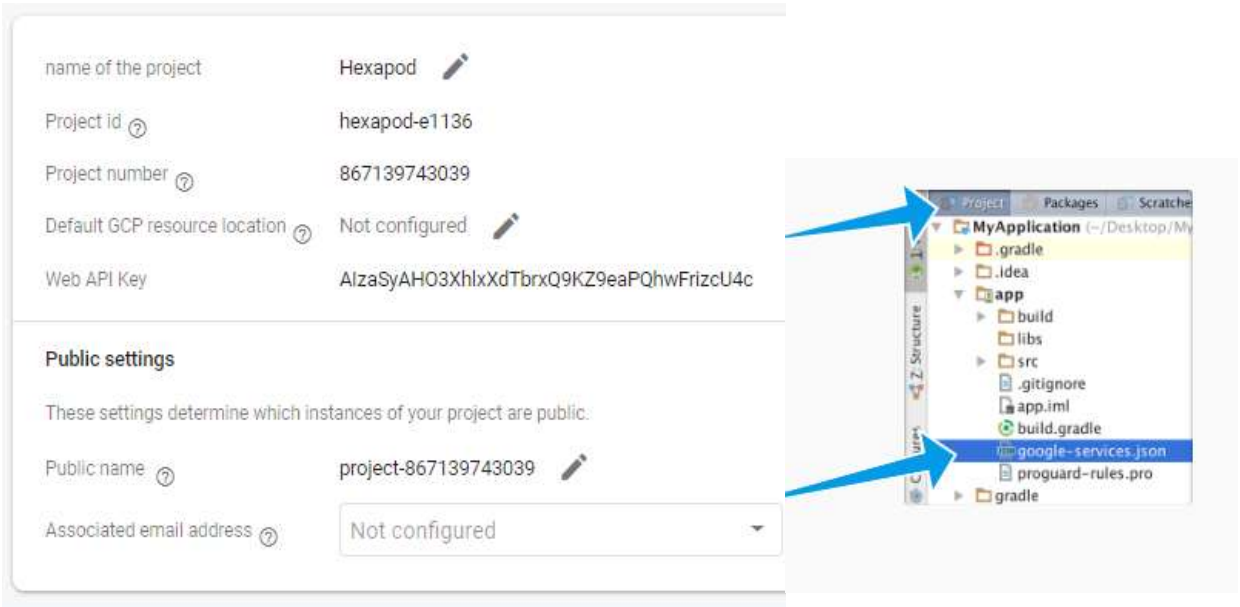


Figure 2. 45: The IP address Firebase.

- Step 5: Edit Our build.gradle files to use the plug-in *"google-service.json"*

Project level build.gradle file (<project>/build.gradle):

```
buildscript {
     repositories { // Check that you have the following
line
               google () // Google's Maven repository }
   dependencies { // Add this line
      classpath 'com.google.gms: google- services: 4.3.3 ' } }
allprojects {
   repositories {//Check that you have the following line
google () // Google's Maven repository} }
```

Application-level build.gradle file (<project>/<app-module>/build.gradle):

```
apply plugin : 'com.android.application' //Add this line
apply plugin : 'com.google.gms.google-services'
dependencies { // Add the Firebase SDK for Google Analytics
implementation ' com.google.firebase: firebase- analytics:
17.2.2 ' }
```

Finally, press "Synchronize" in the bar that appears in the IDE:

Then we are connected to the database successfully !

By call this method, the application can transfer data with Firebase in real time:

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference();
```

2.7.3 IP CAMERA APPLICATION

IP Webcam turns your phone into a network camera with multiple viewing options. View your camera on any platform with VLC player or web browser. Stream video inside WiFi network without internet access. Optional Ivideon cloud broadcasting is supported for instant global access.

For the purpose of remote control and monitoring, the robot must surely transmit image data to the control screen, by using a phone with a camera and available internet connection, the image can be transmitted on long distances. Web server of IP Webcam software with relatively low latency from 4 to 5s, can be accepted.

Figure 2. 46 The logo's IP camera.

2.8 INTRODUCTION TO ANDROID STUDIO

Figure 2. 47: The logo's Android Studio.

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA . On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system.
- A fast and feature-rich emulator.
- A unified environment where you can develop for all Android devices.
- Apply Changes to push code and resource changes to your running app without restarting your app.
- Code templates and GitHub integration to help you build common app features and import sample code.
- Extensive testing tools and frameworks.
- Lint tools to catch performance, usability, version compatibility, and other problems.
- C++ and NDK support.
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine.

2.8.1 PROJECT STRUCTURE

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

- Android app modules
- Library modules
- Google App Engine modules

By default, Android Studio displays your project files in the Android project view, as shown in figure 1. This view is organized by modules to provide quick access to your project's key source files.

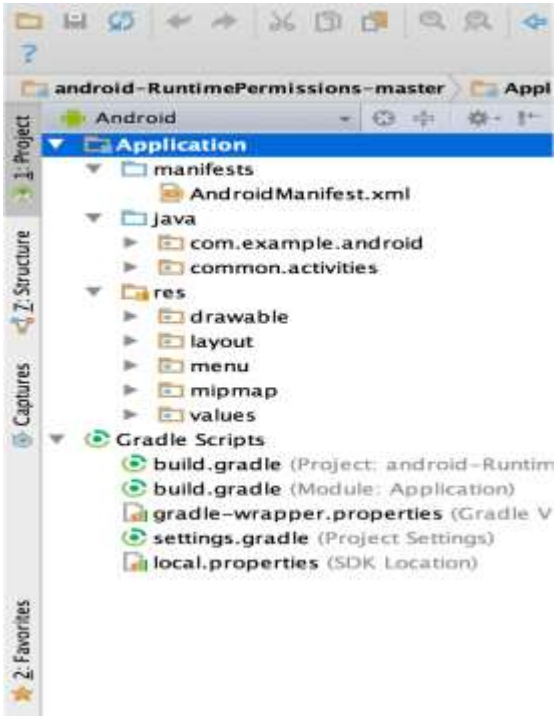


Figure 2. 48: The project files in Android

All the build files are visible at the top level under Gradle Scripts and each app module contains the following folders:

- manifests: Contains the `AndroidManifest.xml` file.
- java: Contains the Java source code files, including JUnit test code.
- res: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select Project from the Project dropdown (in figure 1, it's showing as Android).

You can also customize the view of the project files to focus on specific aspects of your app development. For example, selecting the Problems view of your project displays links to the source files containing any recognized coding and syntax errors, such as a missing XML element closing tag in a layout file.

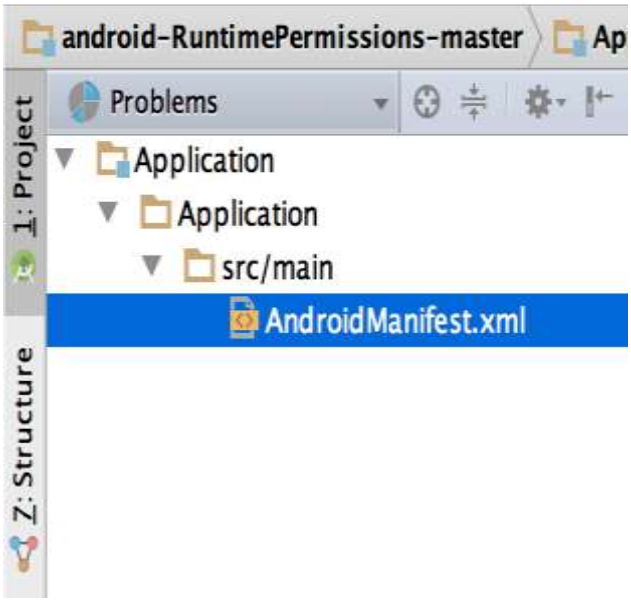


Figure 2. 49:The project files in Problems view, showing a layout file with a problem.

2.8.2 USER INTERFACE

The Android Studio main window is made up of several logical areas identified.

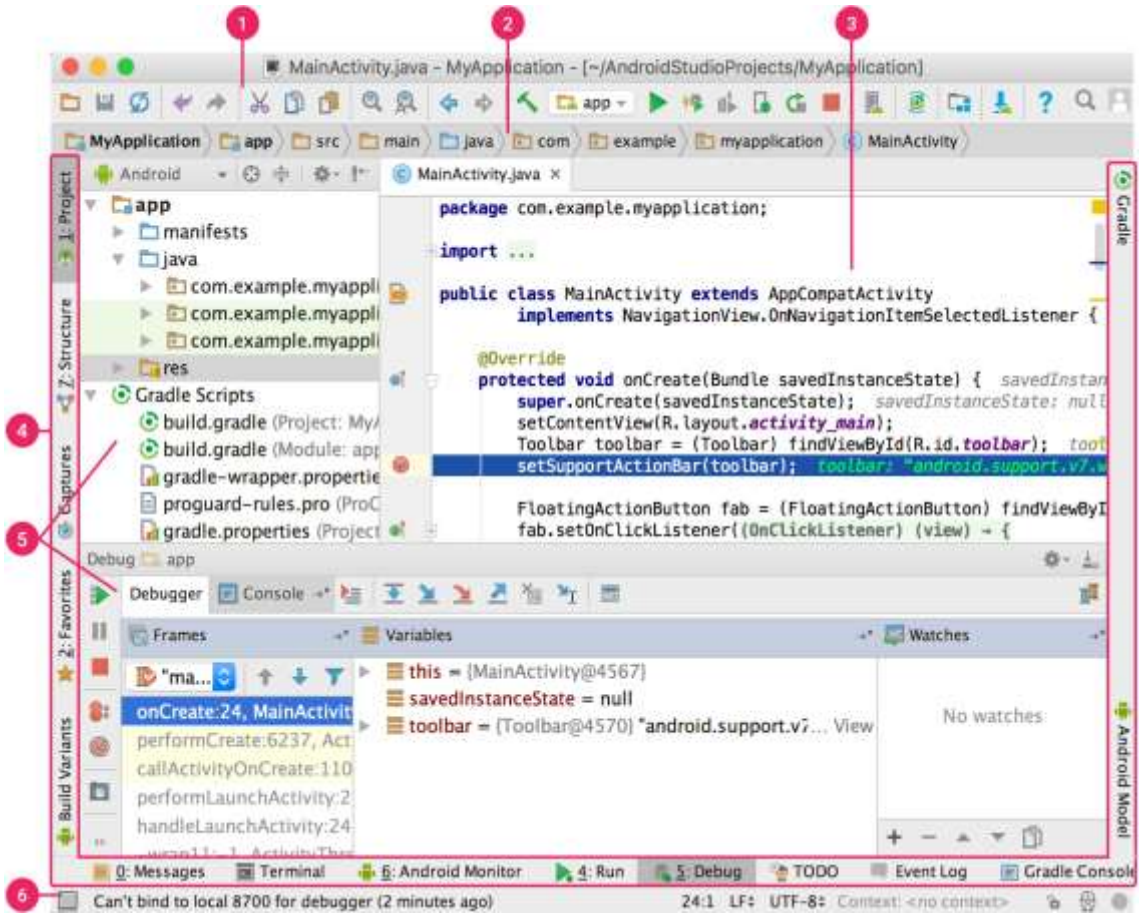


Figure 2. 50: Main Java layout.

1- The toolbar lets you carry out a wide range of actions, including running your app and launching Android tools.

2- The navigation bar helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the Project window.

3- The editor window is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.

4- The tool window bar runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.

5- The tool windows give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.

6- The status bar displays the status of your project and the IDE itself, as well as any warnings or messages.

You can organize the main window to give yourself more screen space by hiding or moving toolbars and tool windows. You can also use keyboard shortcuts to access most IDE features.

At any time, you can search across your source code, databases, actions, elements of the user interface, and so on, by double-pressing the Shift key, or clicking the magnifying glass in the upper right-hand corner of the Android Studio window. This can be very useful if, for example, you are trying to locate a particular IDE action that you have forgotten how to trigger.

2.9 KALMAN FILTER

For the purpose of computational practicality, a discrete-time Kalman filter is used. The implementation will be kept simple since all processing would be done on an Arduino microprocessor with element-wise computational capabilities.

The Kalman filter is an algorithm that uses a series of measurements over time [17] in which case the main measurements are the values obtained from the accelerometer and the gyro sensor. The Kalman filter equation is divided into two groups: the equation for updating the time and the equation for updating the measured value. The time update predictive (time-based) equation uses the present value and covariance estimates to predict the prior estimate for the next moment. The update of the measured value that is responsible for feedback means combining the new value with the prior estimate to correct the posterior estimate. After each step updating the time and updating the measurement, the process is repeated with the previous post-estimation to predict a new prior estimate.

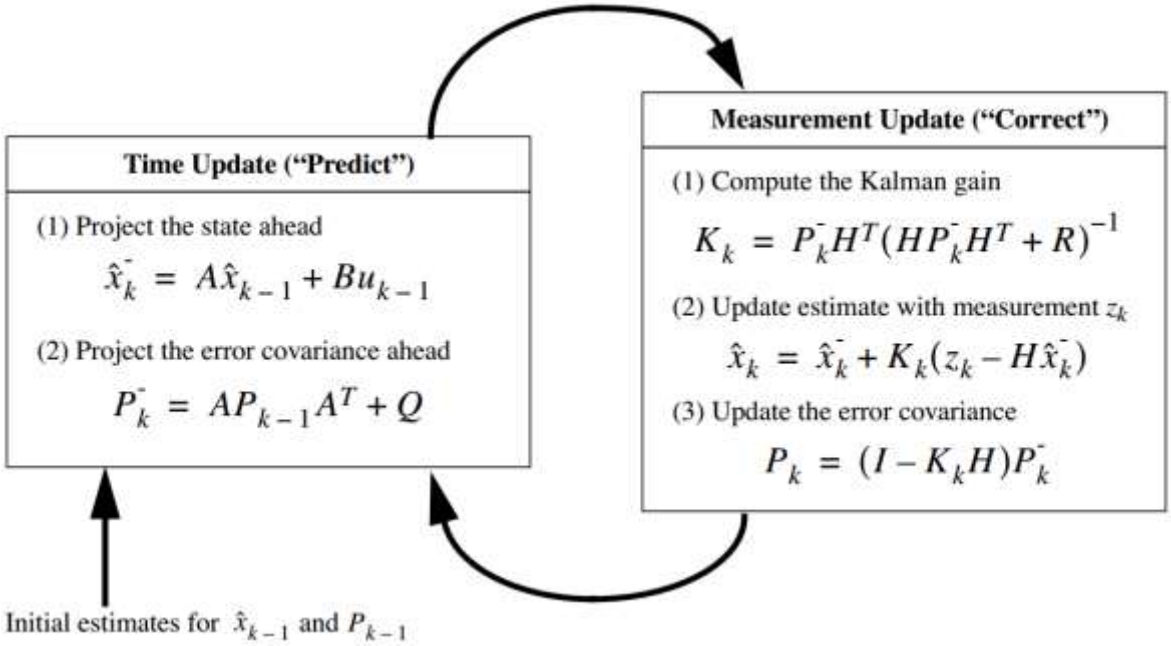


Figure 2. 51: Complete process of Kalman filter.

Supposing we have a simple state having only position and velocity:

$$\vec{x} = \begin{bmatrix} p \\ v \end{bmatrix} \qquad (2.\ 49)$$

What actual position and velocity are, there are range of possible combinations of position and velocity that might be true, but some of them are more likely than others. The Kalman filter assumes both variables (position and velocity) are random and Gaussian distributed. Each variable has a mean value μ, which is the center of the random distribution (and its most likely state), and a variance σ^2, which is the uncertainty. We are estimating a new position based on an old one. The higher the velocity, the more it's likely that the position will be more distant. One measurement tells us something about the others could be. This correlation is captured by a covariance matrix.

Call the estimate at time k: $\hat{x}_k^-$, the known external influences: μ and its covariance matrix P^-_k. To represent the prediction step, use a matrix F. But every state in original estimate could move to a range of states. Each point in $\hat{x}^-_{k-1}$ is moved to somewhere inside a Gaussian blob with covariance Q_k. This produce a new Gaussian blob with a difference covariance but the same mean. Expand the covariance by simply adding Q_k, giving our complete expression for the prediction step:

This is a linear model: states written by vectors, while variables are written in matrices.

$$x^-_k = F.x_{k-1} + B.u_k \qquad (2.\ 50)$$

$$P^-_k = AP_{k-1}A^T + Q_k \qquad (2.\ 51)$$

Where B is the control matrix, u is the control vector. For simple application, these factors can be omitted.

The next step is to refine the estimate with measurements. For the time being, it doesn't matter what sensors measure, probably velocity or position. Each value tells us something indirect about the state. The units and scale of the state we are reading might not be the same units and scale of the state being tracked. H matrix is supposed to be the observation model of the sensors. To represent the distribution of the sensor readings in usual way:

$$\mu = H.\hat{x}_k \qquad (2.\ 53)$$

$$\Sigma = HPH^T \qquad (2.\ 52)$$

In other words sensors are still somewhat unreliable, every state in our original estimate might result in a range of sensor readings. From each reading we observe, we might guess that our system was in a particular state but because there is uncertainty, the covariance of this uncertainty called R_k. The distribution has a mean which is the measurement called z_k.

So now we have two Gaussian blobs: One surrounding the mean of transformed prediction and one surrounding the sensor reading. To find out the most likely state, the two Gaussian blobs are combined together.

Consider as 1D Gaussian bell curve with variance σ^2 and mean μ [5] is defined as:

$$N(x,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\pi^2}} \qquad (2.\ 54)$$

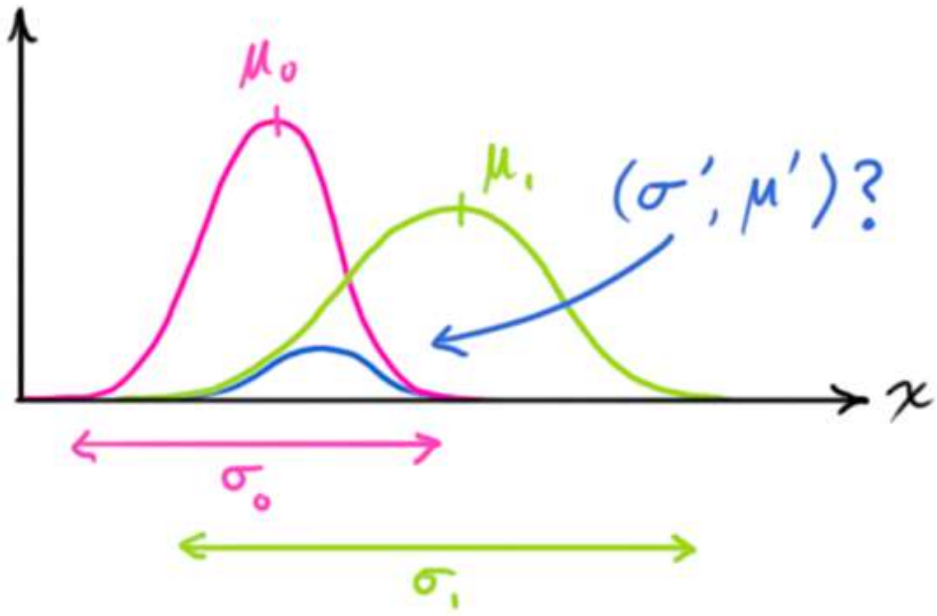


Figure 2. 52: Combination of two Gaussian curves.

$$N(x,\mu_0,\sigma_0).N(x,\mu_1,\sigma_1) = N(x,\mu',\sigma') \qquad (2.\ 55)$$

Substitute equation (2.54) into equation (2.55) to obtain: Combination of two Gaussian curves [18].

$$u' = u_o + \frac{\sigma_0^2(\mu_1-\mu_0)}{\sigma_0^2+\sigma_1^2} \qquad (2.\ 56)$$

$$\sigma'^2 = \sigma_0^2 - K\sigma_0^2 \qquad (2.\ 57)$$

Call[18]: $\mathrm{K} = \dfrac{\sigma_0^2}{\sigma_0^2 + \sigma_1^2}$

$$\mathrm{u}' = \mathrm{u}_0 + \mathrm{K}.(\mu_1 - \mu_0) \quad (2.58)$$

$$\sigma'^2 = \sigma_0^2 - \mathrm{K}\sigma_0^2 \quad (2.59)$$

Re-write (2.58) and (2.59) in matrix form with Σ is the covariance matrix of a Gaussian blob, and u is its mean along each axis:

$$\mathrm{K} = \Sigma_0(\Sigma_0 + \Sigma_1)^{-1} \quad (2.60)$$

$$\begin{aligned} \mu' &= \mu_0 + \mathrm{K}(\mu_1 - \mu_0) \\ \Sigma' &= \Sigma_0 - \mathrm{K}\Sigma_0 \end{aligned} \quad (2.61)$$

K is called the Kalman Gain. Now putting all together, we have two distributions: the predict measurement with $(\mu_0, \sum_0) = (Hx_k, HP_kH^T)$ and the observed measurement with $(\mu_1, \sum_1) = (z_k, R_k)$. Plug these into equation (2.61) to find their overlap [17] :

$$\mathrm{Hx}_k' = \mathrm{Hx}_k + \mathrm{K}(\mathrm{z}_k - \mathrm{Hx}_k) \quad (2.62)$$

$$\mathrm{HP}_k'\mathrm{H}^T = \mathrm{HPH}^T - \mathrm{KHPH}^T \quad (2.63)$$

From (2.59), Kalman gain is: $\mathrm{K} = \mathrm{HPH}^T(\mathrm{HPH}^T + \mathrm{R})^{-1}$ *(2. 64)*

Knock an H off the front of every term in (2.62) and (2.63) (one is hiding inside K), and an H^T off the end of all terms in the equation for P' [17].

$$\mathrm{x}_k' = \mathrm{Hx}_k + \mathrm{K}(\mathrm{z}_k - \mathrm{Hx}_k) \quad (2.65)$$

$$\mathrm{P}_k' = \mathrm{P} - \mathrm{K}'\mathrm{PH} \quad (2.66)$$

$$\mathrm{K}' = \mathrm{PH}^T(\mathrm{HPH}^T + \mathrm{R})^{-1} \quad (2.67)$$

So (2.65), (2.66), (2.67) complete equations of the measurement update step giving x_k' is the new best estimate.

In essence a Kalman filter will have these steps:

- Initialize value X_0, P_0
- Estimate priori state
- Based on the measurement results to correct the estimate
- Repeat

- Implement Kalman filter on C language:

For implementing on an IMU sensor, here the system state at time k:

$$x_k^- = Ax_{k-1} + Bu_k \quad (2.68)$$

Where x_k is the state matrix given by:

$$x_k^- = \begin{bmatrix} \theta \\ \theta_b^{'} \end{bmatrix}_k \quad (2.69)$$

The output of the filter will be the angle θ and the bias $\theta_b^{'}$ (the amount drifted of gyro). We can get the true rate by subtracting the bias from the gyro measurement.

A is the prediction matrix defined as:

$$A = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \quad (2.70)$$

Next is the input control u_k, in this case is angular velocity $(\dot{\theta})$ from gyroscope measurement (deg/s) at time k.

B is the input-control matrix. Since the u_k is the angular velocity and the bias can't be calculated directly then B is defined as:

$$B = \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \quad (2.71)$$

From equation (2.68), (2.69), (2.70), (2.71), the system state is described as follows:

$$\begin{bmatrix} \theta \\ \theta_b^{'} \end{bmatrix}_k = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \theta \\ \theta_b^{'} \end{bmatrix}_{k-1} + \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \cdot \dot{\theta}_k \quad (2.72)$$

$$= \begin{bmatrix} \theta - \theta_b^{'} \Delta t \\ \theta_b^{'} \end{bmatrix}_{k-1} + \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \theta_k^{'} = \begin{bmatrix} \theta + \Delta t(\theta^{'-} \theta_b^{'}) \\ \theta_b^{'} \end{bmatrix} \quad (2.73)$$

Account for noise associated with "world", for instance of tracking a two-wheel robot, the wheels could slip or bump on the ground could slow it down. Model the uncertainty after every prediction step:

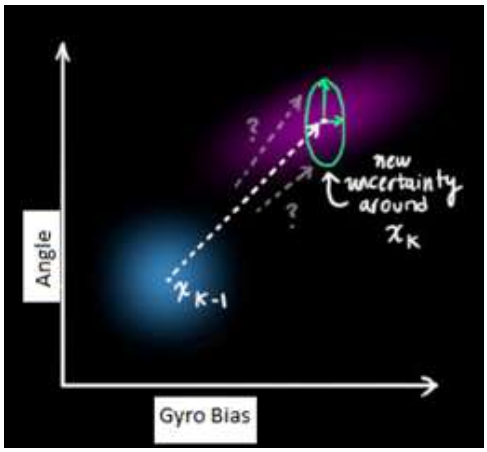


Figure 2. 53: Uncertainty around x_k

Each point in the x_{k-1} is moved to somewhere inside a Gaussian blob with covariance matrix Q_k. In this case, the estimate of bias and accelerometer are considered independently so Q_k is equal to the variance of estimate of the accelerometer and bias. Then Q_k is defined as follows:

$$Q_k = \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\theta_b} \end{bmatrix} .\Delta t \qquad (2.\ 74)$$

The covariance matrix Q_k depends on the current time k, so the variance of the accelerometer and the bias is multiplied by the delta time Δt .

Predict the covariance matrix P_k of x_k by using the following formula:

$$\begin{aligned} \text{Cov}(x_{k-1}) &= P_{k-1} \\ \text{Cov}(Ax_{k-1}) &= AP_{k-1}.A^T \end{aligned} \qquad (2.\ 75)$$

Where A is the prediction matrix (2.70) used above to estimate x_k.

Then we have the estimation of P_k after expanding by adding Q_k [16]

$$P_k^- = AP_{k-1}A^T + Q_k \qquad (2.\ 76)$$

$$\Leftrightarrow \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k-1} \begin{bmatrix} 1 & 0 \\ -\Delta t & 1 \end{bmatrix} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\theta_b'} \end{bmatrix} \Delta t$$

$$= \begin{bmatrix} P_{00} - \Delta t P_{10} & P_{01} - \Delta t P_{11} \\ P_{10} & P_{11} \end{bmatrix}_{k-1} \begin{bmatrix} 1 & 0 \\ -\Delta t & 1 \end{bmatrix} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\theta_b'} \end{bmatrix} \Delta t$$

$$= \begin{bmatrix} P_{00} - \Delta t P_{10} - \Delta t(P_{01} - \Delta t P_{11}) & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} \end{bmatrix}_{k-1} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\theta_b'} \end{bmatrix} \Delta t$$

$$= \begin{bmatrix} P_{00} + \Delta t(\Delta t P_{11} - P_{01} - P_{10} + Q_\theta) & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} + Q_{\theta_b'} \Delta t \end{bmatrix}_{k-1}$$

Next, move to the Correct process [17]:

$$\begin{aligned} y_k &= z_k - Hx_k^- \\ &= z_k - \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \theta_b' \end{bmatrix}_{k-1} = z_k - \theta_{k-1} \end{aligned} \qquad (2.\ 77)$$

- Compute Kalman gain [17]

Estimate error:

$$\begin{aligned} S_k = HP_k^- H^T + R &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_k \begin{bmatrix} 1 \\ 0 \end{bmatrix} + R \\ &= P_{00} + R \end{aligned} \qquad (2.\ 78)$$

Kalman gain: $K_k = P_k^- H^T S_k^{-1}$

$$\begin{bmatrix} K_0 \\ K_1 \end{bmatrix} = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_k \begin{bmatrix} 1 \\ 0 \end{bmatrix} S_k^{-1} \begin{bmatrix} P_{00} \\ P_{10} \end{bmatrix} S_k^{-1} = \frac{\begin{bmatrix} P_{00} \\ P_{10} \end{bmatrix}_k^{\wedge}}{S_k} \qquad (2.\ 79)$$

- Update estimate with measurement z_k [16]

$$\begin{aligned} x_k &= x_k^- - K_k y_k \\ \begin{bmatrix} \theta \\ \theta_b' \end{bmatrix}_k &= \begin{bmatrix} \theta \\ \theta_b' \end{bmatrix}_{k-1} + \begin{bmatrix} K_0 y \\ K_1 y \end{bmatrix} \end{aligned} \qquad (2.\ 80)$$

- Update error covariance:

$$\begin{aligned} P_k &= (I - K_k H) P_k^- \\ \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_k &= \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} K_0 & 0 \\ K_1 & 0 \end{bmatrix}_k \right) \begin{bmatrix} P_{00} & P_{10} \\ P_{01} & P_{11} \end{bmatrix}_k^- \\ &= \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_k^- - \begin{bmatrix} K_0 P_{00} & K_0 P_{01} \\ K_1 P_{00} & K_1 P_{01} \end{bmatrix}_k \end{aligned} \qquad (2.\ 81)$$

CHAPTER 3 THE APP APPLICATION DESIGN AND HARDWARE DESIGN

3.1 APP APPLICATION DESIGN

3.1.1 BOOT SCREEN

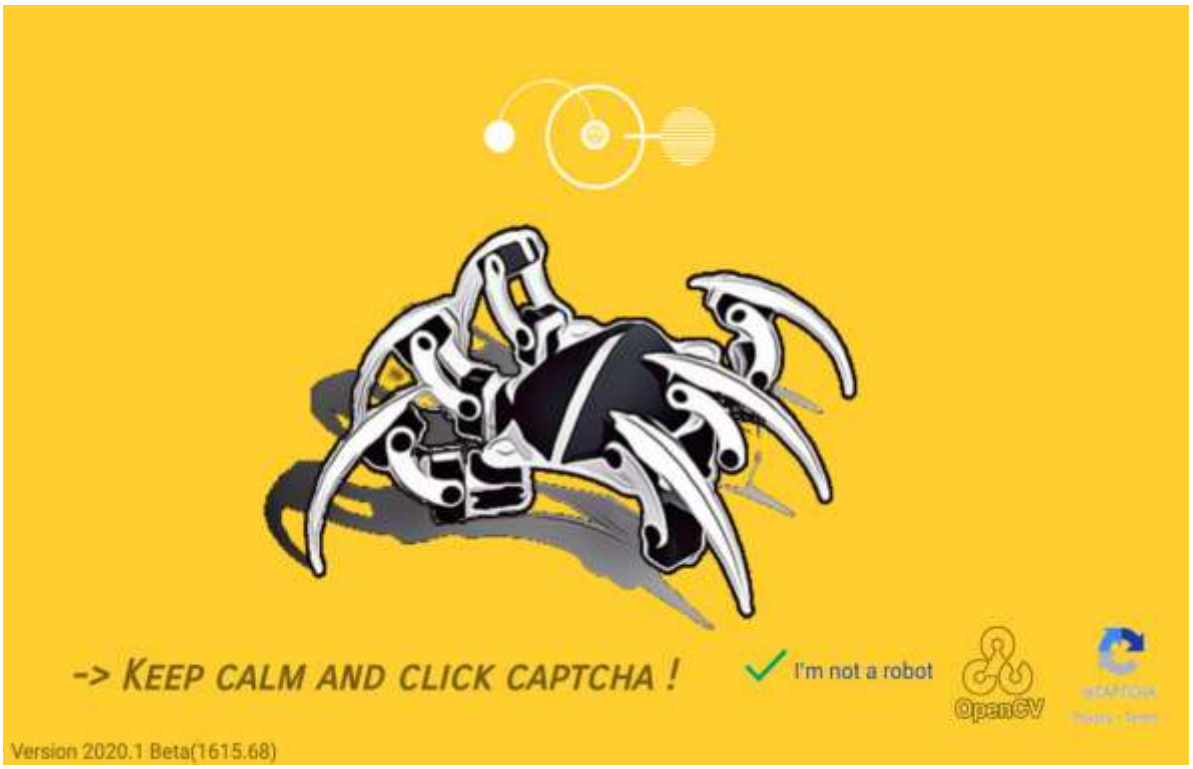


Figure 3. 1: Welcome layout.

3.1.2 CONTROL LAYOUT

Figure 3. 2: The main control layout.

Note:

1. Bluetooth control area.
2. The chosen Bluetooth address.
3. The Bluetooth devices which is available for pairing.
4. The bottom Navigation bar for changing connection Mode.

 -Tracking Mode.

 -Manual Mode.

 -Turn on this device's Bluetooth adapter.

 -Turn off this device's Bluetooth adapter.

 -Start discovering slave device in 120s.

 -Show paring device.

 -Navigation drawer menu.

Figure 3. 3: The members in the same team.

-The Bluetooth Receiving Data Console's.

-The Internet Receiving Data Console's.

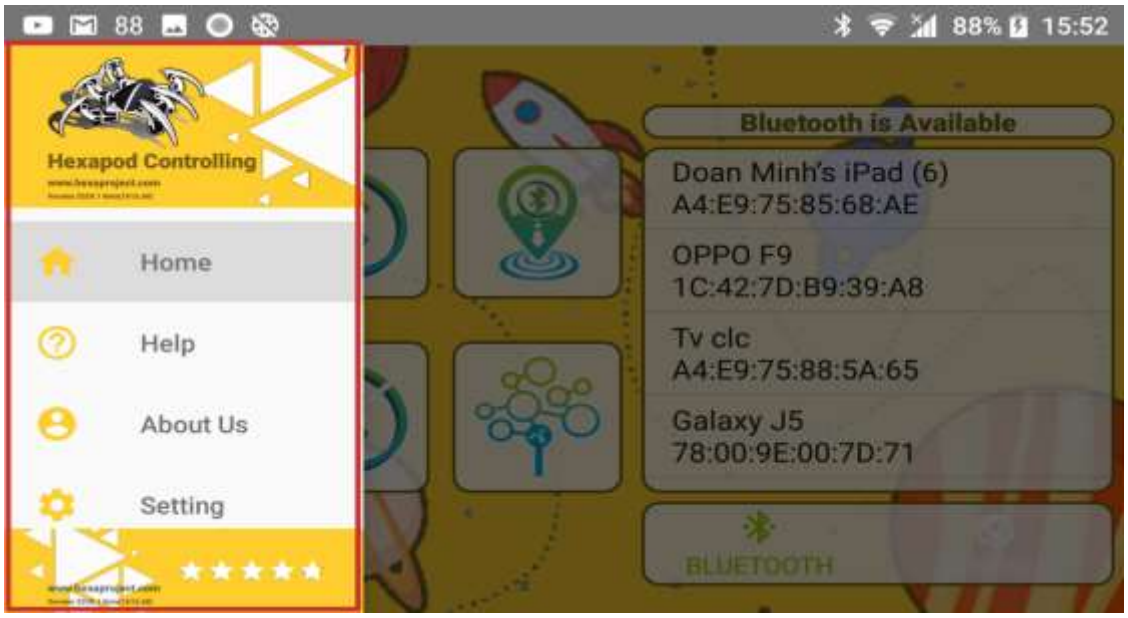


Figure 3. 4: The navigation drawer menu for showing other screen.

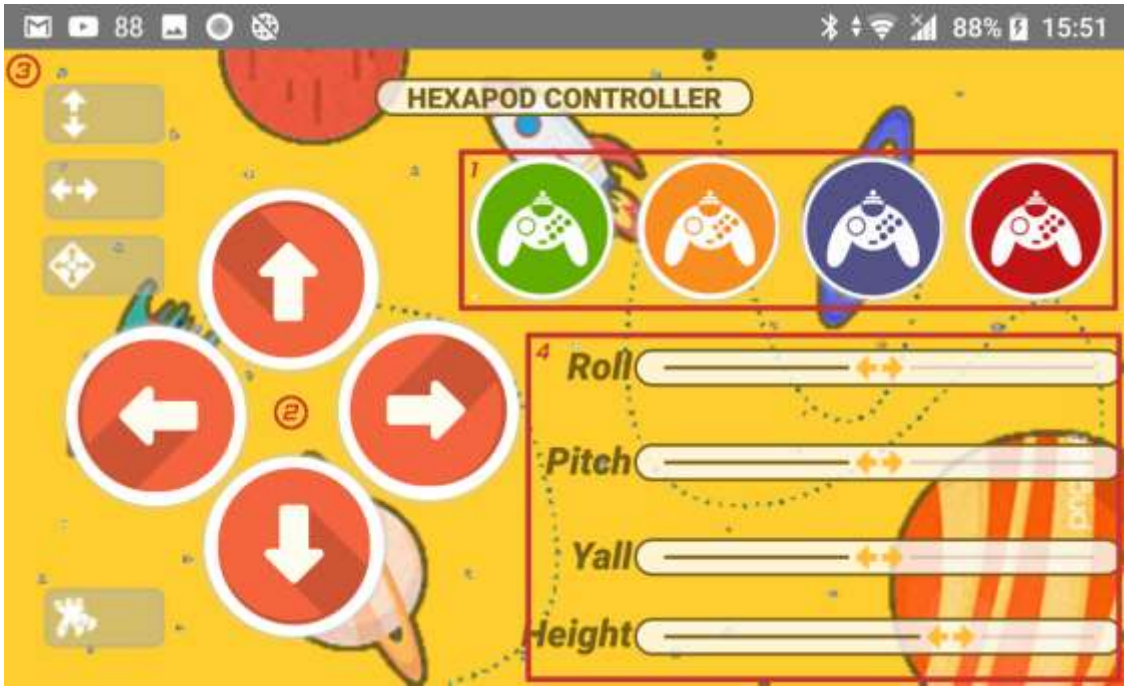


Figure 3. 5: The Bluetooth control layout.

-The Bluetooth Console's opened when click this button on main screen.

1. Function button: Preset movements for Hexapod.
2. Basic navigation motion of Hexapod
3. Basic motion mode Switch
4. Three basic Hexapod's rotation angle showing by Seek-bar

-The Tracking mode Console's opened when click this button on main screen.

Figure 3. 6: The camera phone layout.

1. Touch coordinates: Where your finger is on screen.
2. RGB color values at this coordinates.
3. HSV color values at this coordinates in range $^{H}[0, 179]$, $^{S}[0, 255]$, $^{V}[0, 255]$.
4. HSV color values at this coordinates in range $^{H}[0, 360]$, $^{S}[0, 1]$, $^{V}[0, 1]$.
5. Function switch.
6. Controlling switch.
7. Capture screen for image processing.

Figure 3. 7: The Internet control layout.

-The Internet Console's opened when click this bottom button on main screen.

1. Basic navigation motion of Hexapod.

2. Data read from real-time database.

3. Seek-bar for controlling camera.

4. Function switch.

-Start/Stop

-Up/Down

-Balancing Mode

-The Internet Receiving data Console's opened

When click this bottom

Button on main screen.

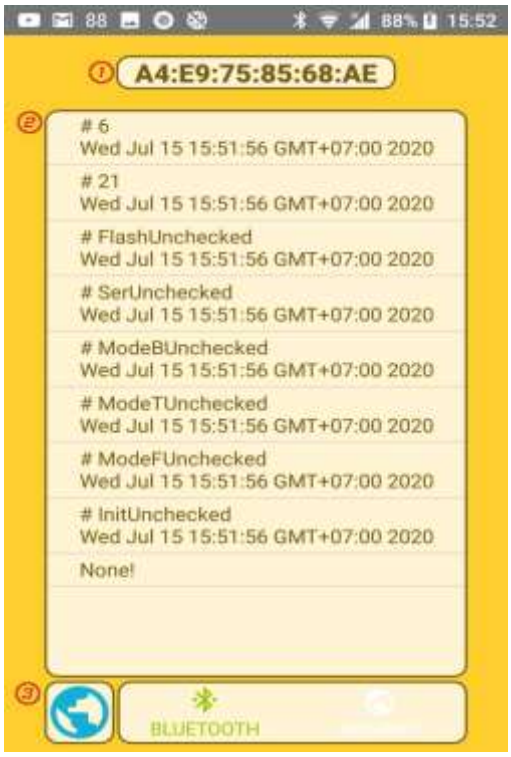


Figure 3. 8: The monitoring data layout.

1. The Bluetooth address that this devices is connecting with.

2. Data read from real-time database.

3. Connection Status bar.

3.2 HARDWARE DESIGN

3.2.1 MICROPROCESSOR

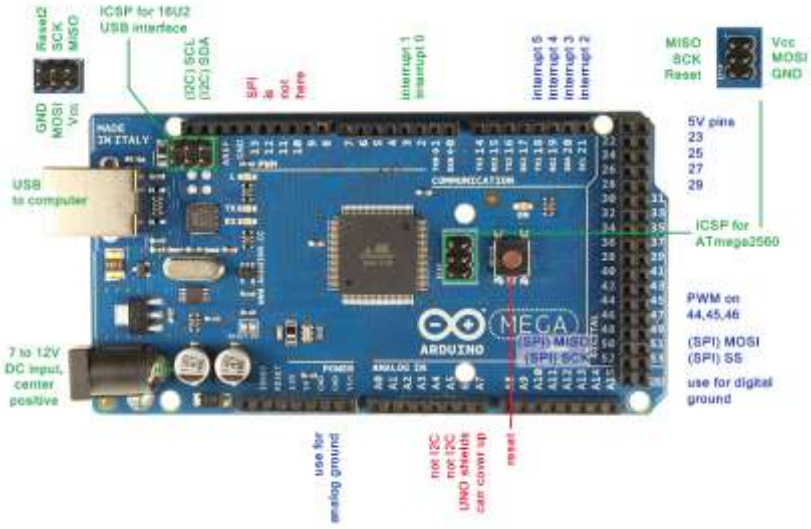


Figure 3. 9: Board Arduino Mega 2560

Arduino mega 2560 board is based on the ATMega2560 chip from Atmel's 8bit mega AVR microprocessor. Enables the programming of complex control applications that are equipped with powerful ROM, RAM and Flash memory modules, digital I/O outlets, multiple terminals capable of outputting PWM signals, Analog signals and communication standards such as UART, SPI, TWI (I2C).

Table 3. 1: Technical specifications for board Arduino Mega 2560.

Microcontroller	**ATMega2560**
Operating voltage	5V
Clock speed	16MHz
Input voltage (recommend)	7V-12V
Input voltage (limit)	6V – 20V
Digital I/O pins	54 (15 PWM outputs)
Analog pins	16
DC current per I/O pin	20mA
DC current for 3.3V pin	50mA
Flash memory	256KB
SRAM	8KB
EEPROM	4KB

- Power source for Arduino: There is a DC jack to power the board (7-12V), in addition to a Vin also can be used to power the Arduino. If use a 5V external power source, 5V pin can also be used to power Arduino
- PWM pulse feet: There are 15 PWM digital pins (pins 2-13, pins 44, 45, 46) in threads using pins 44 and 45 for right and left motors.
- I2C communication: 2 pins Digital I2C communication is 20 (SDA) and 21 (SCL).
- Serial communication: Supports up to 4 Serial ports 0, 1, 2, 3 (Pin order 0, 1, 19, 18, 17, 16, 15, 14). Here we use Serial 3 to communicate with Bluetooth module HC05.
- Interrupts: There are 6 external interrupt pins 0, 1, 2, 3, 4, 5 (order of pins are 2, 3, 21, 20, 19, 18). Here we use interrupt 0, 1 to read 2 channels A of the two encoders of the left and right motor.

3.2.2 POWER SUPPLY

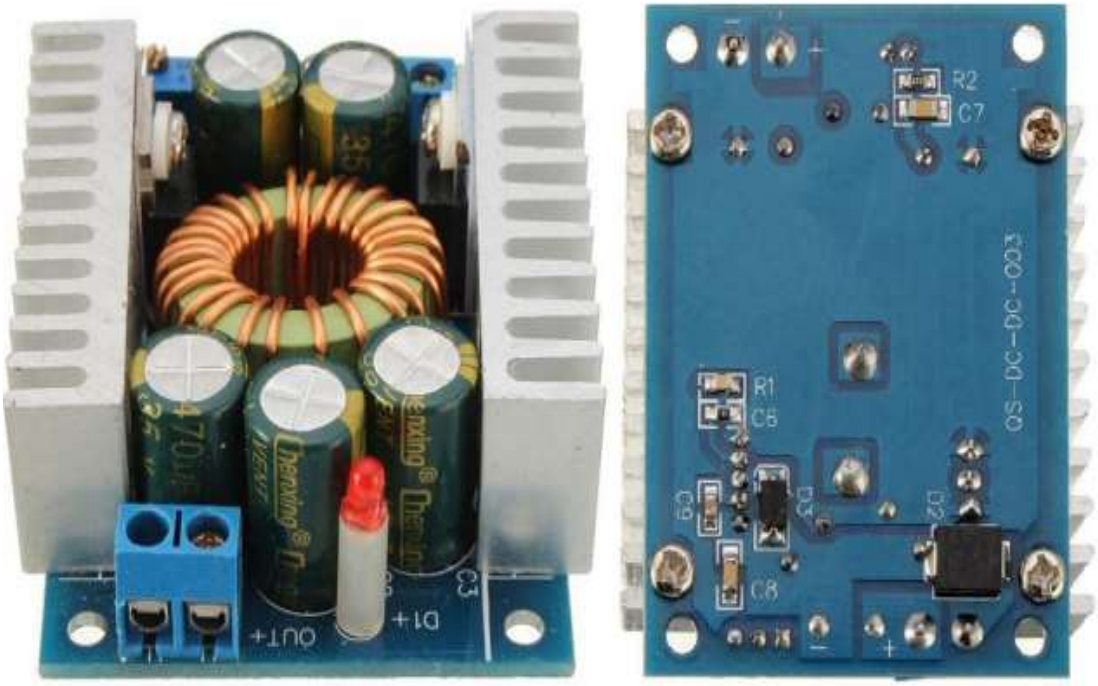


Figure 3. 10: Module Buck DC-DC 12A.

In this projects, Arduino MEGA does not have integrated 3.3V and 5V power circuits, so it is difficult to supply standard 5V or 3.3V for modules or sensors. On the other hand, the power supply for the robot is a fixed 12V power source, so to supply 5V to the driver controller and processor, we have to use Module BUCK DC - DC - 12V to solve this problem.

Table 3. 2: Technical specifications for module Buck DC-DC 12A.

Specifications	Module BUCK DC-DC 12A
Module Type	12A Step down
Input Voltage	4.5-30V
Output voltage	1.25-30V adjustable
Output current	0-12A 100W
Operating temperature	-40 to + 85 degrees
Operating frequency	300 kHz
Conversion efficiency	up to 95%
Module dimensions	60 mm x 51 mm x 22 mm

Features:

- Adjustable Voltage CV.
- Short circuit protection: current limitation 14A, Fuse.
- Overheating protection: Yes (for temperature, automatic shutdown after disconnecting the output) .
- Input reverse polarity protection: No, (if required in the input line to the diode) .
- Module dimensions: 60 mm x 51 mm x 22 mm (L x W x H).

3.2.3 MODULE BLUETOOTH

Figure 3. 11: Module bluetooth HC05.

Module Bluetooth HC05 offers bandwidth up to 720 Kbps in data range from 10m to 100m. Unlike infrared (IrDA), when the communication needs to surface the device close to each other, bluetooth uses multidirectional radio waves that allow transmission through non-metallic obstacles. Bluetooth transmits at 2.4GHz frequency and uses continuous scalar broadcast technology.

Table 3. 3: Specification for module HC05.

Module	Bluetooth HC-05
Power	3.3-5V
Current	30mA when paring, after paring the normal transmission operation 8mA
Baud rate	1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
Operating frequency range	2.4GHz
Bluetooth protocol	Bluetooth Specification v2.0 EDRo
Dimension	28mm x 15mm x 2.35mm

- Setting default:
- Baud rate: 9600, N, 8, 1.
- Pairing code: 1234

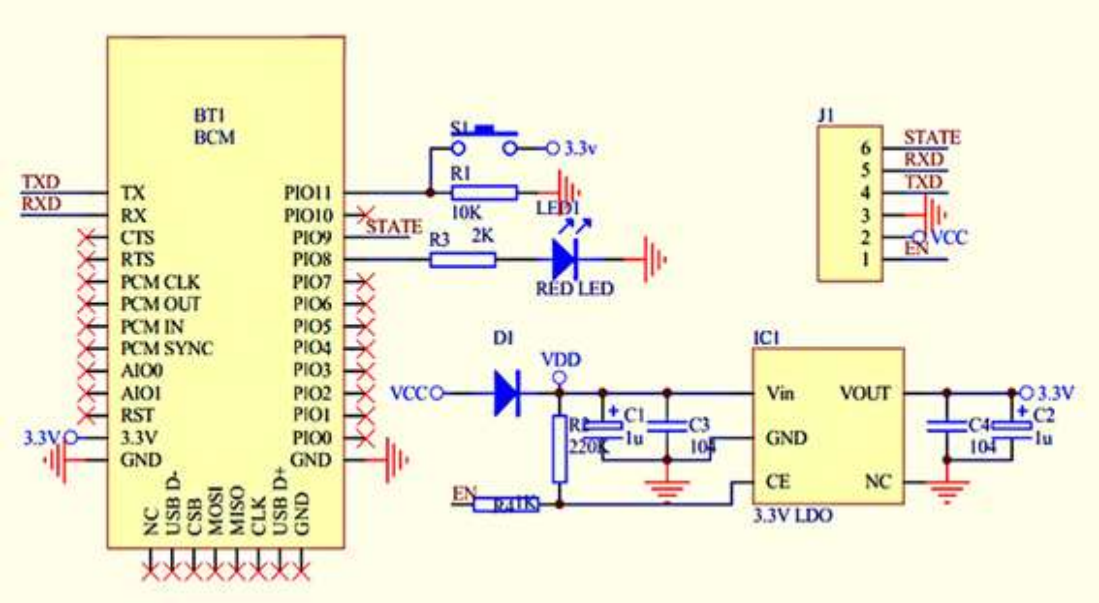


Figure 3. 12: HC05 schematic.

3.2.4 MPU6050 SENSOR

The MPU6050 is one of the Inertial Measurement Unit (IMU) sensors - a combination of two Accelerometer sensors and a Gyroscope sensor. Today, the IMU is one of the most commonly used sensors in electronic devices. They can be found in smartphones, handheld controls, electronic devices worn on people ... IMU sensors help us determine the state of an object attached to sensors in three dimensions.

Figure 3. 13: MPU 6050 – Accelerometer and Gyroscope module.

The module also have two auxiliary pins which can be used to interface external IIC modules like an magnetometer, however it is optional. Since the I2C address of the module is configurable more than one **MPU6050 sensor** can be interfaced to a Microcontroller using the AD0 pin. This module also has well documented and revised libraries available hence it's very easy to use with famous platforms like Arduino. So if you are looking for a sensor to control motion for your RC Car, Drone, Self balancing Robot, Humanoid, Biped or something like that then this sensor might be right choice for you.

Table 3. 4: Pins of module MPU6050.

Module	GY521 – MPU6050
VCC	5VDC power supply
GND	Ground
Serial Clock (SCL)	I2C clock pin
Serial Data (SDA)	I2C data communication pin
Auxiliary Serial Data (XDA)	Data pin (connected to other sensors)
Auxiliary Serial Clock (XCL)	Clock pulse pin (connect to other sensors)
AD0	Bit 0 of I2c address
Interrupt (INT)	Interrupt pin to indicate that data is available for MCU to read.

The hardware of the module is very simple, it actually comprises of the MPU6050 as the main components as shown above. Since the module works on 3.3V, a voltage regulator

is also used. The IIC lines are pulled high using a 4.7k resistor and the interrupt pin is pulled down using another 4.7k resistor.

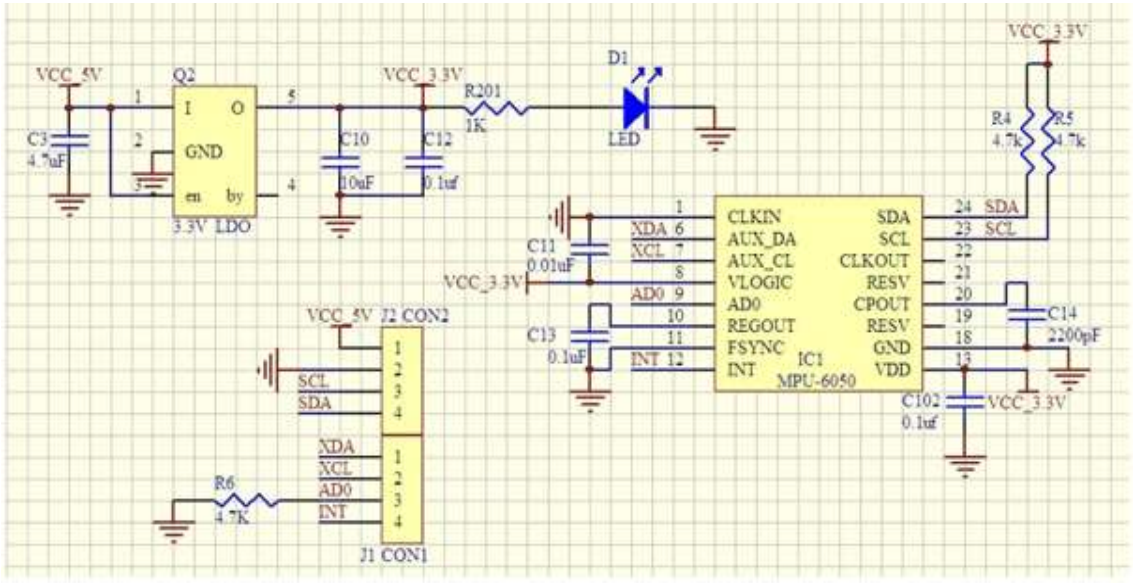


Figure 3. 14: The schematic diagram.

- Within the framework of the use of the default I2C address and no external interrupts and other auxiliary sensors, XDA, XCL, INT, AD0 pins will not be used.
- GY521 uses the I2C protocol:
- I2C communication standard:
- I2C is a two-wire interface consisting of a data line (SDA) and a single wire (SCL). In I2C, a device can be Master or Slave. Each slave has a fixed address, Master has to access corresponding address to call that slave.
- In communication with the Arduino, MPU6050 in Slave mode, the maximum bus speed is 400 Khz.
- To program the I2C interface, we use Arduino's Wire.h library, which writes the I2C.h subroutine to read and export I2C data.
- Configuration of MPU6050 registers:
- Register WHO_AM_I: Contains the default slave address value of the MPU value of 0x68. The least significant bit of the MPU-60X0's I2C address is determined by

the value of the AD0 pin. The address can be changed by AD0 pin. (The address would be changed to 0x69 if AD0 is connected to 5V pin).

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
75	117	-	WHO_AM_I[6:1]						-

- Sample Rate divider: Calculates the sampling rate of the MPU

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
19	25	SMPLRT_DIV[7:0]							

This register specifies the divider from the gyroscope output rate used to generate the Sample Rate for the MPU-60X0. The sensor register output, FIFO output, and DMP sampling are all based on the Sample Rate. The Sample Rate is generated by dividing the gyroscope output rate by SMPLRT_DIV: Sample Rate = Gyroscope Output Rate / (1 + SMPLRT_DIV)

Set the register value to 0x07, specifying a sampling rate of 1 Khz when the DLPF is disabled (DLPF_CFG = 0 or 7) and 8kHz when the DLPF is enabled.

Configuration Register: Configure synchronous sampling frame and the Digital Low Pass Filter for sensor. The default value is 0x00.

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1A	26	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		

Gyroscope Configuration Register: Configures the angular velocity range, where the value 0x00 is selected with a range from -250 to 250 degrees / s.

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1B	27	XG_ST	YG_ST	ZG_ST	FS_SEL[1:0]		-	-	-

FS_SEL	Full Scale Range
0	± 250 °/s
1	± 500 °/s
2	± 1000 °/s
3	± 2000 °/s

Accelerometer Configuration Register: Measure the angular acceleration range, here select 0x00 with range from -2g to 2g.

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1C	28	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]		-		

AFS_SEL	Full Scale Range
0	± 2*g*
1	± 4*g*
2	± 8*g*
3	± 16*g*

Power Management register: select the clock source for the MPU, the default clock source is the 8 MHz quartz oscillator.

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6B	107	DEVICE _RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		

Data Gyro Record Registers: store the angular velocity values of the three axes.

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
43	67	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT[7:0]							

Acc Data Record Registers: store the angular acceleration values of the three axes.

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT[7:0]							

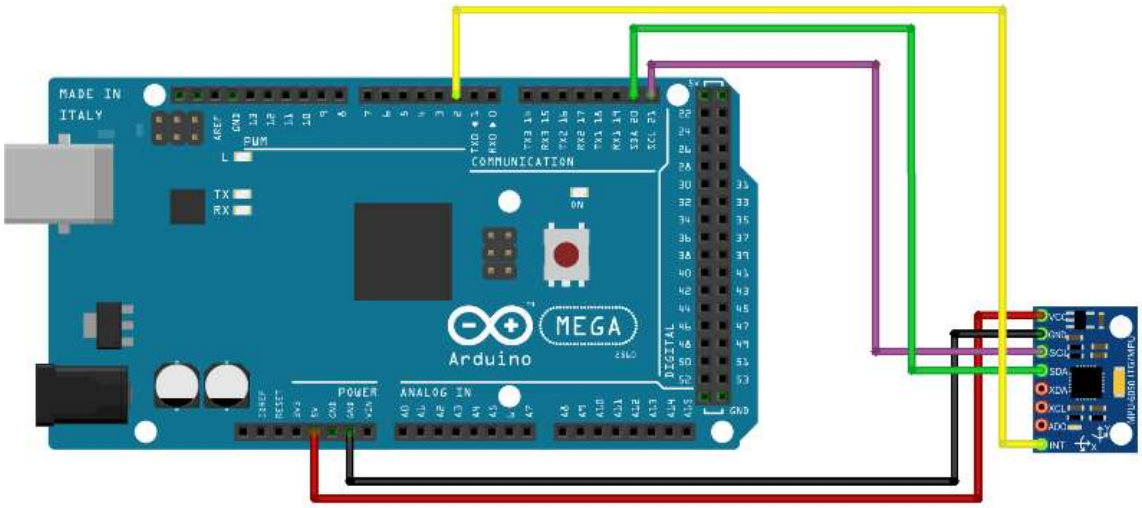


Figure 3. 15: The diagram Wiring GY521 with Arduino.

Most of Servo motors operates from 4.8V to 6.5V, the higher the voltage higher the torque we can achieve, but most commonly they are operated at +5V. Almost all standard servo motors can rotate only from Either 0° to 180° due to their gear arrangement so make sure the project can live with the half circle if no or 0° to 360° motor or modify the motor to make a full circle. The gears in the motors are easily subjected to wear and tear, in this case, application requires stronger and long running motors so we choose servo motor with metal gears.

Next comes the most important parameter, which is the torque at which the motor operates. Again there are many choices here but let us assume the one with 2.5kg/cm torque which comes with the MG996R Motor. This 2.5kg/cm torque means that the motor can pull a weight of 2.5kg when it is suspended at a distance of 1cm. So if we suspend the load at 0.5cm then the motor can pull a load of 5kg similarly if you suspend the load at 2cm then can pull only 1.25. Based on the load which you use in the project you can select the motor with proper torque. The below picture will illustrate the same.

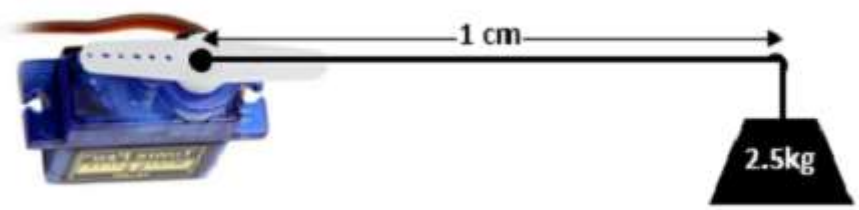


Figure 3. 16: Servo mini SG90.

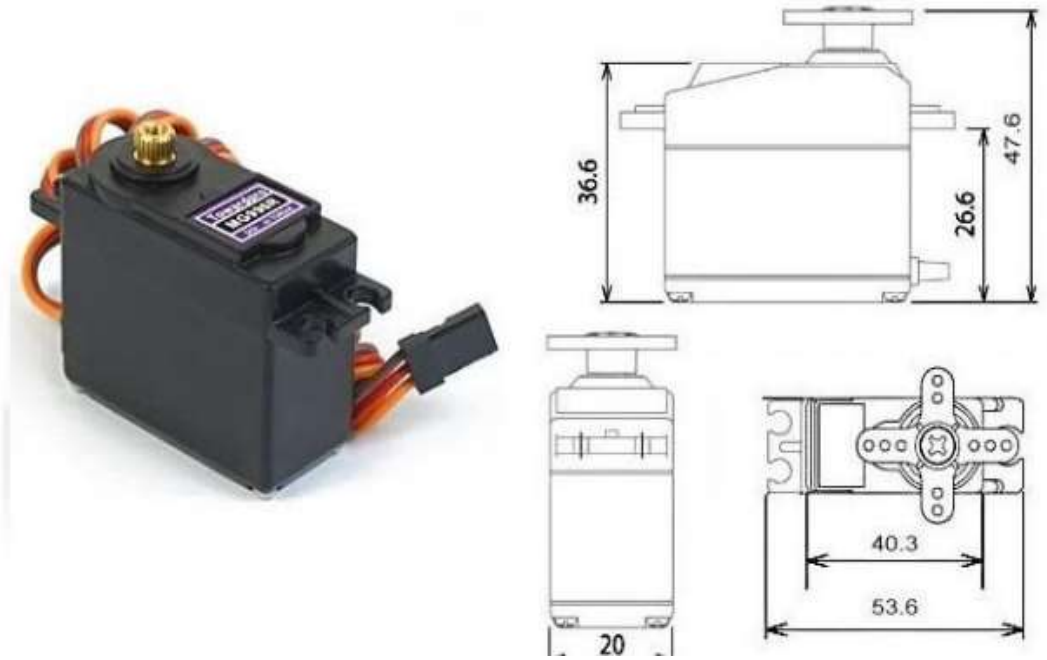


Figure 3. 17: MG996R High Torque Metal Gear Dual Ball Bearing Servo.

The MG996R is a metal gear servo motor with a maximum stall torque of 11 kg/cm. Like other RC servos the motor rotates from 0 to 180 degree based on the duty cycle of the PWM wave supplied to its signal pin.

Table 3. 5: Specification for MG996R.

Specifications	ATMega2560
Operating Voltage	4.8 V - 7.2 V
Running Current	500 mA – 900 mA (6V)
Stall Current	2.5 A (6V)
Stall torque	.4 kg/cm (4.8 V), 11 kg/cm (6 V)
Dead band width	5 µs
Operating speed	.17 s/60º (4.8 V), 0.14 s/60º (6 V)
Temperature range	0º - 55º
Weight	55 g
Dimension	40.7 x 19.7 x 42.9 mm approx
Gear Type	Metal
Rotation	0° - 180°

From the figure below we can understand that the PWM signal produced should have a frequency of 50Hz that is the PWM period should be 20ms. Out of which the On-Time can vary from 1ms to 2ms. So when the on-time is 1ms the motor will be in 0° and when 1.5ms the motor will be 90°, similarly when it is 2ms it will be 180°. So, by varying the on-time from 1ms to 2ms the motor can be controlled from 0° to 180°.

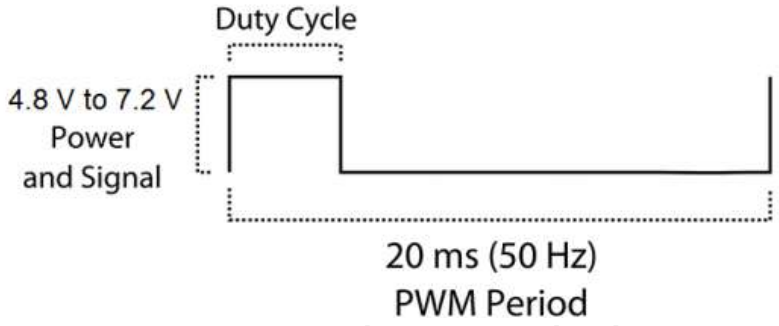


Figure 3. 18: The PWM signal cycle.

Table 3. 6: The PCA9685 pinout configuration.

Wire Number	Wire Colour	Description
1	Brown	Ground wire connected to the ground of system
2	Red	Powers the motor typically +5V is used
3	Orange	PWM signal is given in through this wire to drive the motor

3.2.5 CHANNEL SERVO DRIVER

Driving servo motors with the Arduino Servo library is pretty easy, but each one consumes a precious pin - not to mention some Arduino processing power. The Adafruit 16-Channel 12-bit PWM/Servo Driver will drive up to 16 servos over I2C with only 2 pins. The on-board PWM controller will drive all 16 channels simultaneously with no additional Arduino processing overhead. What's more, you can chain up to 62 of them to control up to 992 servos - all with the same 2 pins.

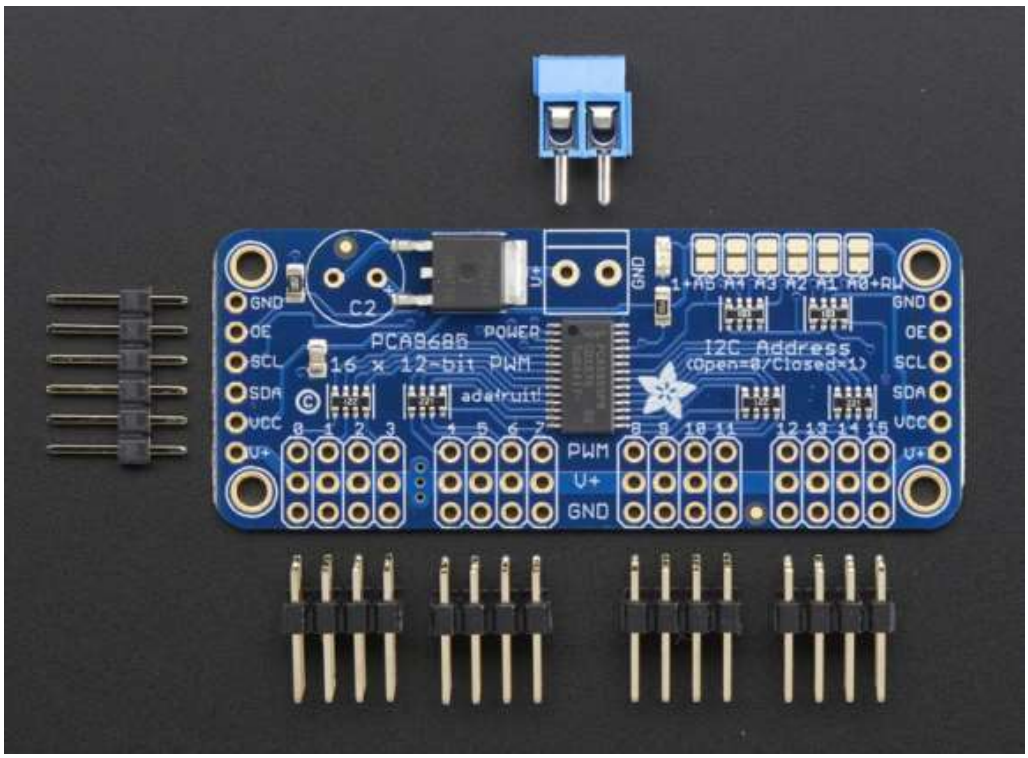


Figure 3. 19: The PCA pinout.

Table 3. 7: Specification for PCA9685.

Module	PCA9685
GND	The power and signal ground pin
VCC	Logic power pin (should be 3 - 5V max)
V+	Power to the servos (5 - 6VDC)
Serial Clock (SCL)	I2C clock pin
Serial Data (SDA)	I2C data communication pin
OE (Output Enable)	Can be used to quickly disable all outputs
Output ports	There are 16 output ports. Each port has 3 pins: V+, GND and the PWM output. Each PWM runs completely independently but they must all have the same PWM frequency

Connecting a Servo came with a standard 3-pin female connector that will plug directly into the headers on the Servo Driver. Align the plug with the ground wire (usually black or brown) with the bottom row and the signal wire (usually yellow or white) on the top.

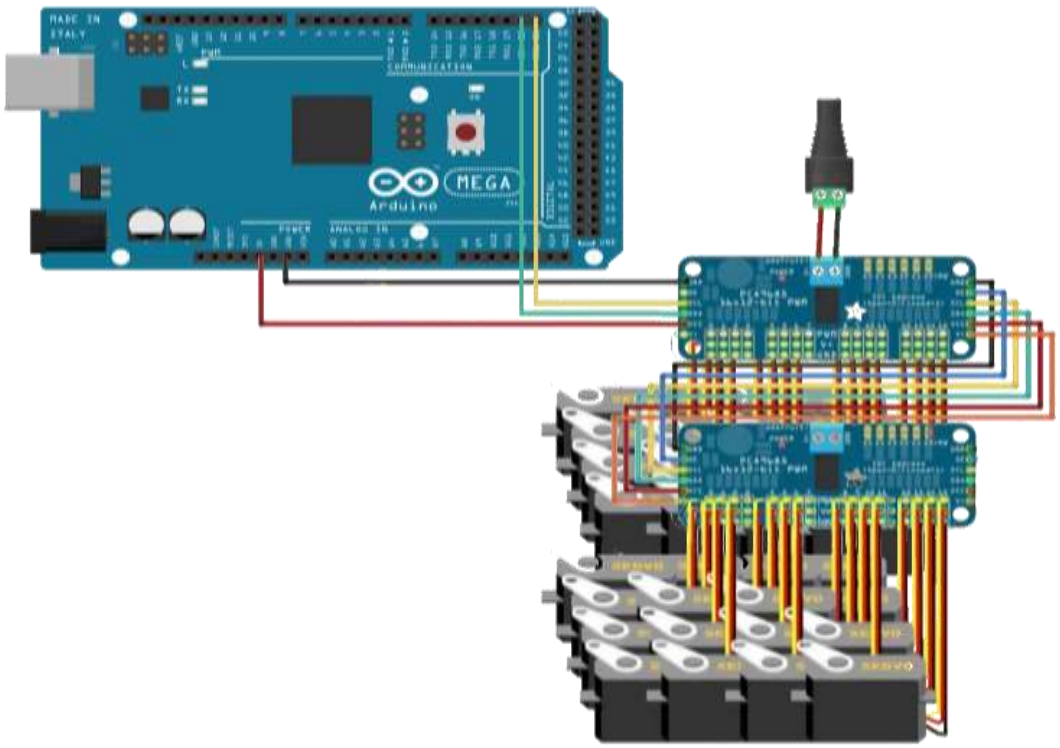


Figure 3. 20: The Wiring Arduino and the PCA9685 pinout.

Chaining Multiple Drivers (up to 62) can be chained to control still more servos. With headers at both ends of the board, the wiring is as simple as connecting a 6-pin parallel cable.

The I2C base address for each board is 0x40. The binary address that you program with the address jumpers is added to the base I2C address. To program the address offset, use a drop of solder to bridge the corresponding address jumper for each binary '1' in the address.

In this project, we use 2 parallel board to control 18 servo (9 - 9) so we have to declare address:

- Board 0: Address = 0x40 Offset = binary 00000 (no jumpers required) .
- Board 1: Address = 0x41 Offset = binary 00001 (bridge A0).

```
#include <Wire.h>

#include <Adafruit_PWMServoDriver.h>

Adafruit_PWMServoDriver pwm1 = Adafruit_PWMServoDriver(0x40);
```

3.2.6 MATRIX LED 8x8

Matrix led 8x8 is a Led board of 8 rows and 8 columns (total of 64 leds). If controlled directly by Arduino, that is not feasible. Therefore, IC MAX7219 was born to control countless LEDs with only 5 pins.

Figure 3. 21: The led matrix 8x8 module

Table 3. 8: Specification for Led matrix 8x8.

Specifications	Matrix led 8x8
Operating Voltage	DC 4.7V – 5.3V
Running Current	320mA
Temperature range	0^{o}C- 50^{o}C
Dimension	32mm X 32mm X 8 mm
Show column style	Cathode Row

A bit is a basic information unit of a computer (Binary digIT). The possible bit values are 0 and 1 (true / false). We will use it to control the LEDs. Understandably, if bit is 1 then the light is on and the 0 light is off.

When there are many lights, it will form a bit sequence. For example, if there are 3 lights 101 then they will turn on - off - light on. Because Led Matrix consist of many lights, using each bit becomes more complicated. After Bit is Byte, Byte is also a unit of information of a computer. For each byte we get 8 bits, or in other words, 1 byte = 8 bits. Using bytes we will be able to control 8 lights at the same time.

The value of the byte will start with the letter B followed by a binary string. Because each byte consists of 8 bits, matrix LEDs are often designed to be multiples of 8.

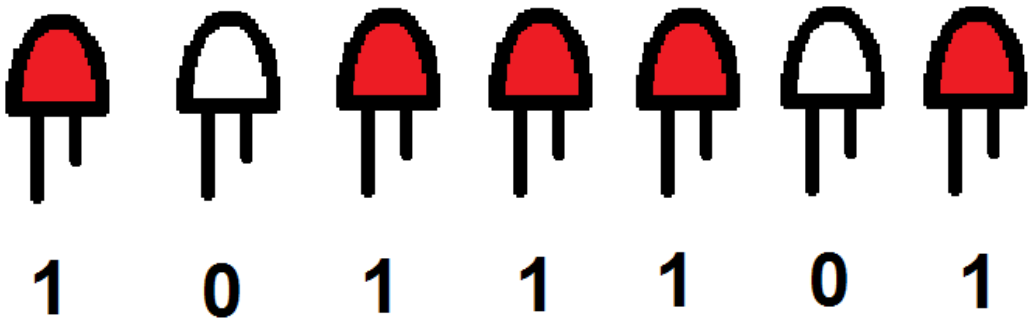


Figure 3. 22: A byte - control

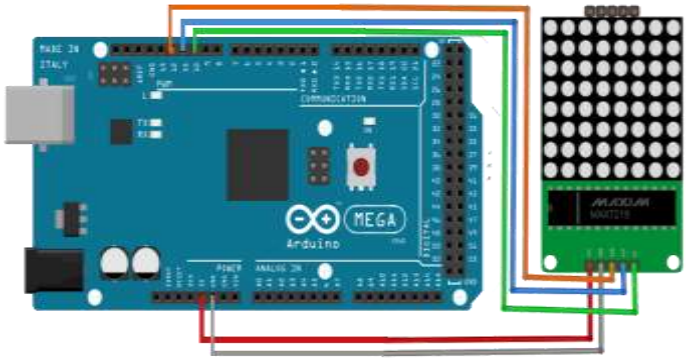


Figure 3. 23: Wiring Arduino and Led matrix 8x8.

Table 3. 9: Specification for wiring Led matrix.

Pinouts	ATMega2560
Vcc	5V
GND	GND
Din (Serial data in)	12
CLK (Clock)	11
CS (Load)	10

In this project we has written some animations to display on the 8x8 matrix led depending on the movement of the robot as follows:

Movement	Hex value
Led_right	0x38,0x14,0x8A,0xC5,0xC5,0x8A,0x14,0x38
Led_left	0x1C,0x28,0x51,0xA3,0xA3,0x51,0x28,0x1C
Led_Forward	0x18,0x24,0x5A,0xA5,0xC3,0x81,0x18,0x3C

Led_Backward	0x3C,0x18,0x81,0xC3,0xA5,0x5A,0x24,0x18
Standby	1- 0x00,0x00,0x00,0x18,0x18,0x00,0x00,0x00 2- 0x00,0x00,0x3C,0x24,0x24,0x3C,0x00,0x00 3- 0x00,0x7E,0x42,0x42,0x42,0x42,0x7E,0x00 4- 0xFF,0x81,0x81,0x81,0x81,0x81,0x81,0xFF
ON	0x81,0x5A,0x3C,0x18,0xFF,0x3C,0x42,0x81
Balance	1- 0x00,0x00,0x00,0x18,0x18,0x00,0x00,0x00 2- 0x00,0x00,0x18,0x24,0x24,0x18,0x00,0x00 3- 0x00,0x18,0x24,0x42,0x42,0x24,0x18,0x00 4- 0x18,0x42,0x24,0x81,0x81,0x24,0x42,0x18 5- 0x81,0x42,0x00,0x00,0x00,0x00,0x42,0x81 6- 0x81,0x00,0x00,0x00,0x00,0x00,0x00,0x81

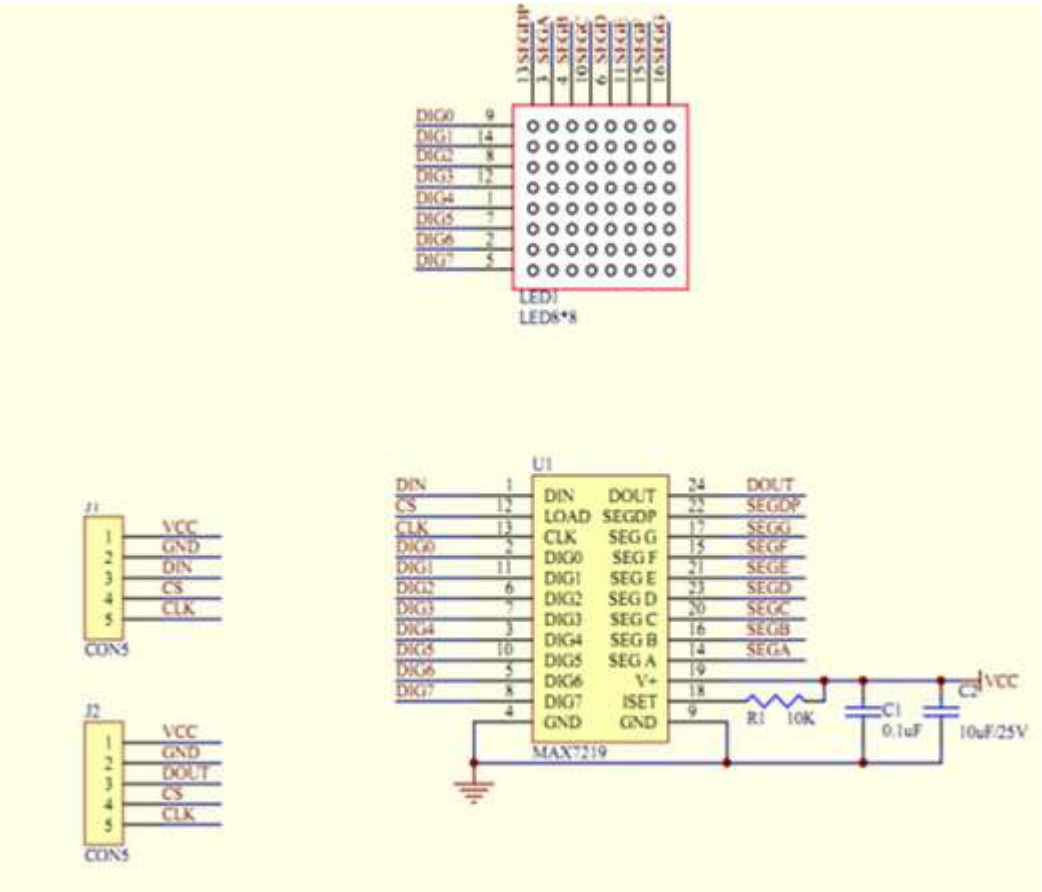


Figure 3. 24: Led matrix 8x8 schematic diagram.

3.2.7 GRAPHIC DESIGN FOR FRAME ROBOT

Figure 3. 25: CorelIDRAW graphics suite 2020.

CorelDraw is a vector graphics software commonly used in graphic design - has similar features to Illustrator, allowing the use of built-in tools to create different vector objects, creating works. design.

Graphics are divided into 2 types: bitmap and vector. With bitmap, it is managing objects by pixels (pixels - photos taken by camera, phone) or image files exported as jpeg, jpg, png. And vector graphics are the use of coloring algorithms based on the limits of lines to design an image. Compared to bitmaps, images created by vector graphics allow easy zooming without compromising image quality.

CorelDraw is considered as one of the graphics software with a huge data warehouse with clipart system, high quality digital images, professional temples, diverse fonts ... will help designers create drawings. 2D, 3D applications in many different fields: construction, architecture, fine arts, ...

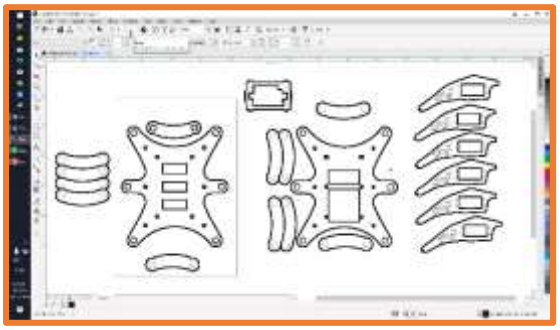
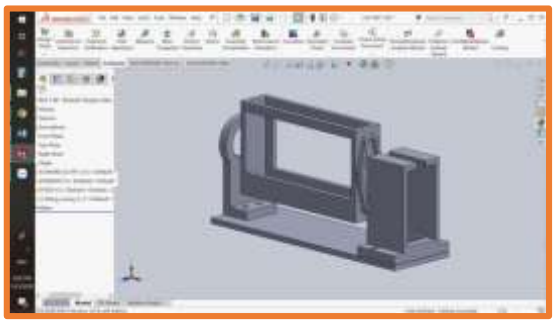
Figure 3. 26: SOLIDWORK student edition 2016.

SolidWorks CAD software is a product of SolidWorks Company. This brand has been released since 1997 by Dassault Systèmes SolidWorks Corp (The company is a subsidiary of Dassault Systèmes, S. A. in Vélizy, France)..

Already a technical person, everyone knows that CAD (Computer Aided Design). In general terms, CAD means using computers in the process of sketching, 3D modeling, assembling and publishing drawings.

In the traditional method, technical drawings are often drawn by hand. This job requires a lot of effort and time, especially the intricate details. Therefore, today CAD is widely used in many fields, not only in the field of manufacturing mechanics but also in construction, architecture, fine arts, commerce, medicine ... In this article , we will reduce the scope of our engineering design activities. Currently, SolidWorks is used quite popular in the world. In our country, this is a widely used software not only in the field of mechanics, but it is also expanded to other fields such as application science, electricity, simulation mechanics, ..

Table 3. 10: The designing software.

		The entire Robot's parts are made up of Mica material, including the body, legs and electrical circuit frame. To design these mica parts, we use The vector graphics editor - CorelDRAW, the Robot's parameters are mentioned at the beginning of chapter 1.
	SOLIDWORKS	To get an overview of the Robot's moving mechanism, Team has designed the 3d modeling through Solidworks, to check the structures and dimensions before kinematic simulation with the algorithm via Matlab software.
	SOLIDWORKS	For the purpose of designing the device that attaches the phone to the top of the robot for image processing, this element is designed and exported in .stl format used by 3D printers, in deep blue color, 2h printing time.

In addition, the software from SolidWork also provides users with the best features of designing 3D parts, assembling those parts to form parts of machines and publish

drawings. 2D details that are very popular features of SolidWorks software, in addition to other features such as: Dynamic analysis (Simulation), Dynamic analysis (Motion).

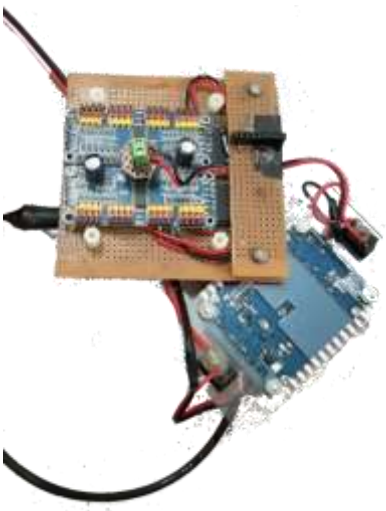
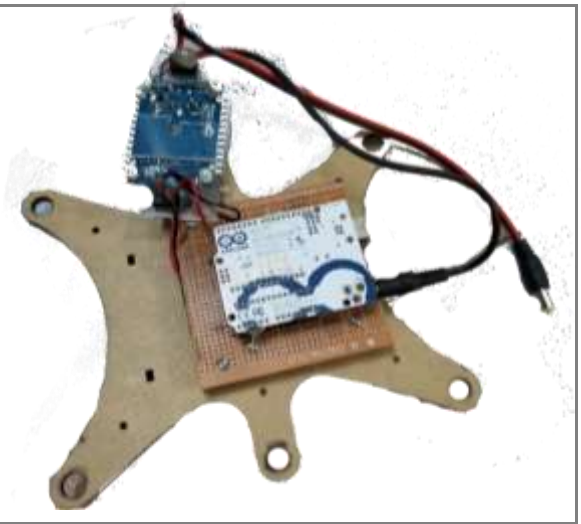
3.2.8 HARDWARE CONTRUCTION

Figure 3. 27: Build the Servo Bracket Assemblies.

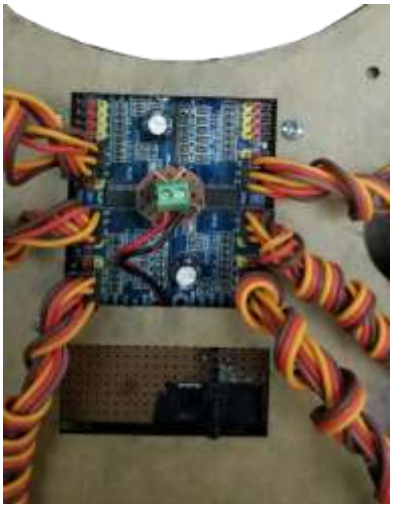
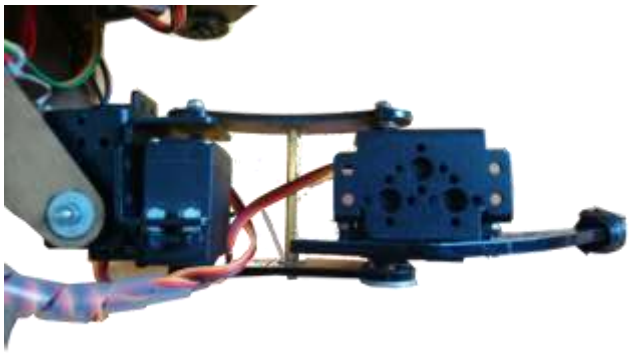
With the servo brackets built into pairs of mirrored joints, we can now attach the servos themselves to the brackets. The orientation of the servos is not as difficult to figure out as the orientation of the brackets, but it can still be a touch confusing.

We are now in the final stretch of the build process for our Hexapod robot. In this step we will attach all of the previously-assembled legs to the upper chassis plate (the one to which we attached all the electronics) and the auxiliary controlling circuit board to Robot's body. When attaching the coxa servos on the legs to the servo horns on the upper chassis plate, the servos must be oriented so that the leg faces directly away from the center point of the robot. As we discussed when assembling the legs, in order for the robot firmware to operate correctly, the robot must be assembled so that when the servos are centered, the robot is in a predetermined posture.

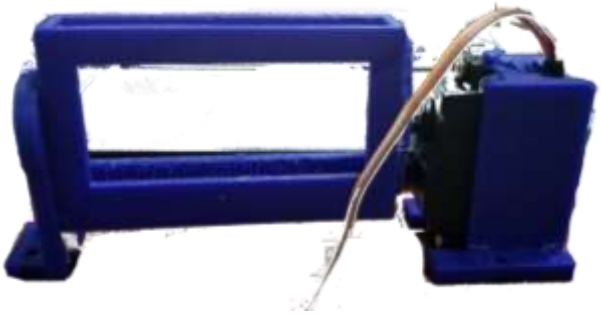
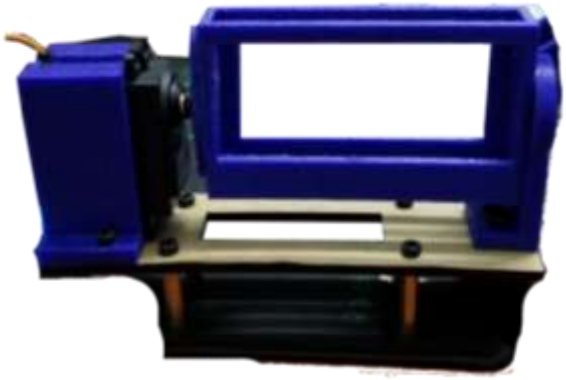
Table 3. 11: Separate parts of hardware construction.

	
PCA 9685 circuit board.	PCA 9685 circuit board underneath.

.

<table>
<tr><td>  </td><td>  </td></tr>
<tr><td>The upper side of PCA 9685 circuit board.</td><td>Matric led circuit and bluetooth layout.</td></tr>
<tr><td> 
</td><td>  </td></tr>
<tr><td>Lower body construction.</td><td>Wiring PCA 9685 circuit borad with servos.</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Leg mechanism</td><td>Leg mechanism</td></tr>
</table>

.

	
Smartphone holder mechanism	Smartphone holder mechanism

CHAPTER 4 THE ALGORITHM

4.1 CONTROL ROBOT FLOWCHART

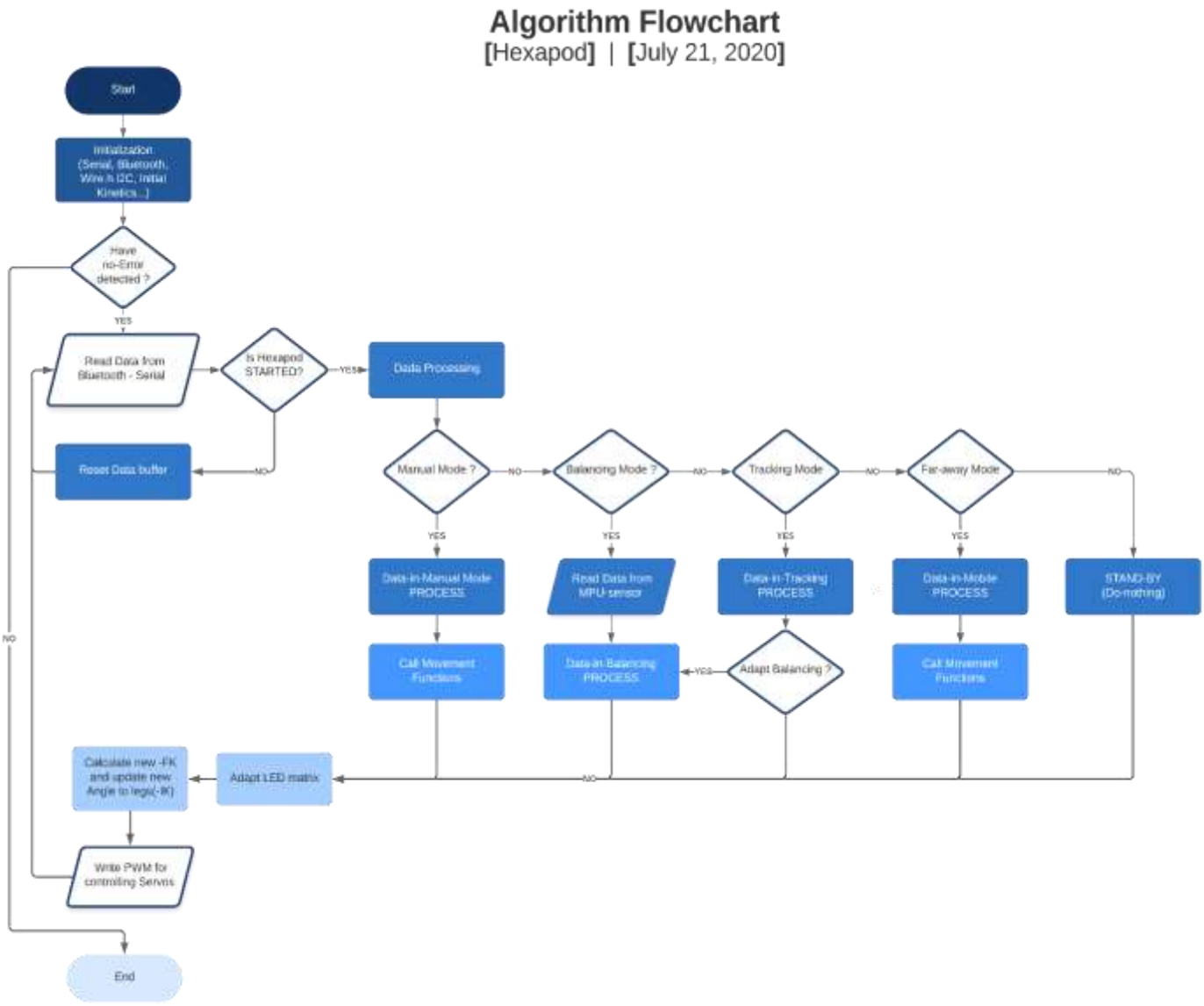


Figure 4. 1: The algorithm flowchart for moving.

The operation of the robot corresponds to the rotation of the servo. The operation of multiple servos at the same time will help the robot move. Depending on needs, we will transmit data wirelessly to robots and make requests from operators. The control procedure is shown in the following figure 4.1.

The operation of the robot corresponds to the rotation of the servo. The operation of multiple servos at the same time will help the robot move. Depending on needs, we will transmit data wirelessly to robots and make requests from operators. The control procedure is shown in the following figure 4.1.

Initially, we started the robot system to create a signal communication between devices. The signals generated are serial communication, Bluetooth signal, I2C communication, robot kinetics.

The second step is to receive the signal from the phone. Check what signal is received. If the receiving robot is allowed to move, the third step is to check which signal is executed including: manual mode, balancing mode, tracking mode, Far-away mode.

Control mode	**Manual mode**	**Balancing mode**	**Tracking mode**	**Far-away mode**
Feature	Call function is corresponding to : • Move forward • Move backward • Move turn left • Move turn right	- Read data from MPU sensor to adapt balancing algorithm to robot.	-Read data receive from smartphone under image processing. -Tracking object by moving toward this object.	-We transmit data via Firebase -Communicate with robot through internet.

Corresponding to each operating mode, the matrix LED will display the corresponding status. The fourth step is to calculate the rotation angle by using the inverse kinematic for the robot. Depending on the type of movement, the application of inverse dynamics will vary.

The fifth step is to create a rotation angle for the servos by calling the PWM in the Arduino program. After the function is called, the robot will move.

4.2 GENERAL FLOWCHART OF IMAGE PROCESSING:

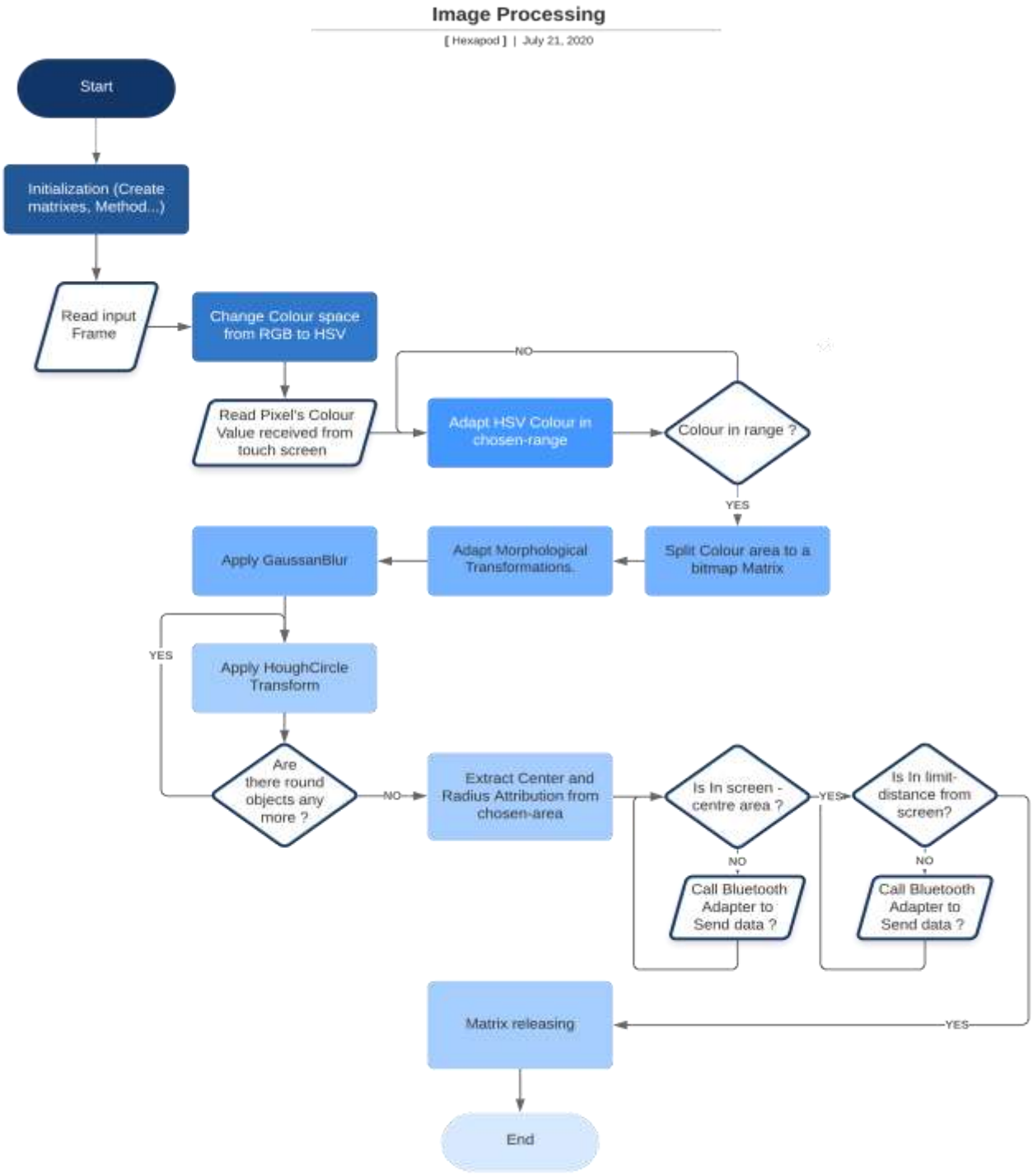


Figure 4. 2: The algorithm flowchart for image processing.

Image processing will serve for object tracking. Using the support library in image processing is OpenCV. The input data is the RGB image matrix. The library supports inputFrame.rgba() function to capture each frame and processing on that frame.

OpenCV support us function inputFrame.rgba() for capturing each frame and processing on that frame. The output of this function is the RGB image which has three

color channels: Red, Green, Blue. However, rgba also consist of the fourth channel which provides us (a), light adjustment parameter. We convert RGB to HSV color space by using:

*Imgproc.cvtColor(inputArray **src**, OutputArray **dst**, int **code**, int **dstCn**)*

- **src**: input image: 8-bit unsigned, 16-bit unsigned or single-precision floating-point.
- **dst**: output image of the same size and depth as src.
- **code**: We use Improc.COLOR_RGB2HSV.
- **dstCn**: number of channels in the destination image.

We use 3: H, S and V channels. The output of converting RGB to HSV is filtered by

*Core.inRange(inputArray **src**, InputArray **lowerb**, inputArray **upperb**, OutputArray **dst**)*

This function filters the image by scanning all matrices based on the range value. Pixels in that range are made as 255 (maximum of 8 bits image - white). On the other hand, they are made as 0 (minimum of 8 bits image – black).

- **src**: first input array.
- **lowerbs**: inclusive lower boundary array or a scalar.
- **upperb**: inclusive upper boundary array or a scalar.
- **dst**: output array of the same size as src.

After that, we apply Morphology transform to get rid of the small details in the image by using this function:

*Imgproc.morphologyEx(InputArray **src**, OutputArray **dst**, int **op**, InputArray **kernel**)*

- **src:** Source image. The number of channels can be arbitrary. CV_8U.
- **dst:** Destination image of the same size and type as src.
- **op**: MORPH_OPEN - an opening operation, MORPH_CLOSE - a closing operation.
- **kernel**: Structuring element. We use a kernel mask MORPH_RECT, Size(5,5).

As the image after processing still has a lot of noise and in order to avoid mistakes in the identifying process, the image should be blurred with function

*Imgproc.GaussianBlur(inputArray **src**, OutputArray **dst**, Size **ksize**, double **sigmaX**, double **sigmaY**).*

- **src**: input image. The image can have any number of channels.
- **dst**: output image of the same size and type as src.
- **ksize**: Gaussian kernel size with ksize.width and ksize.height.
- **sigmaX**: Gaussian kernel standard deviation in X direction.
- **sigmaY**: Gaussian kernel standard deviation in Y direction.

Finally, we apply Hough circle transform to find whether or not having round shape objects in this image:

Imgproc.HoughCircles(InputArray image, OutputArray circles, int method, double dp, double minDist, double param1=100, double param2=100, int minRadius=0, int maxRadius=0)

- **image**: 8-bit, single-channel, grayscale input image.
- **circles:** Output vector of found circles. Each vector is encoded as a 3-element floating-point vector $(x, y, radius)$.
- **method**: Detection method to use.
- **dp**: Inverse ratio of the accumulator resolution to the image resolution.
- **minDist**: Minimum distance between the centers of the detected circles.
- **param1:** First method-specific parameter. *Canny edge detector()*
- **param2:** Second method-specific parameter.
- **minRadius**: Minimum circle radius.
- **maxRadius**: Maximum circle radius.

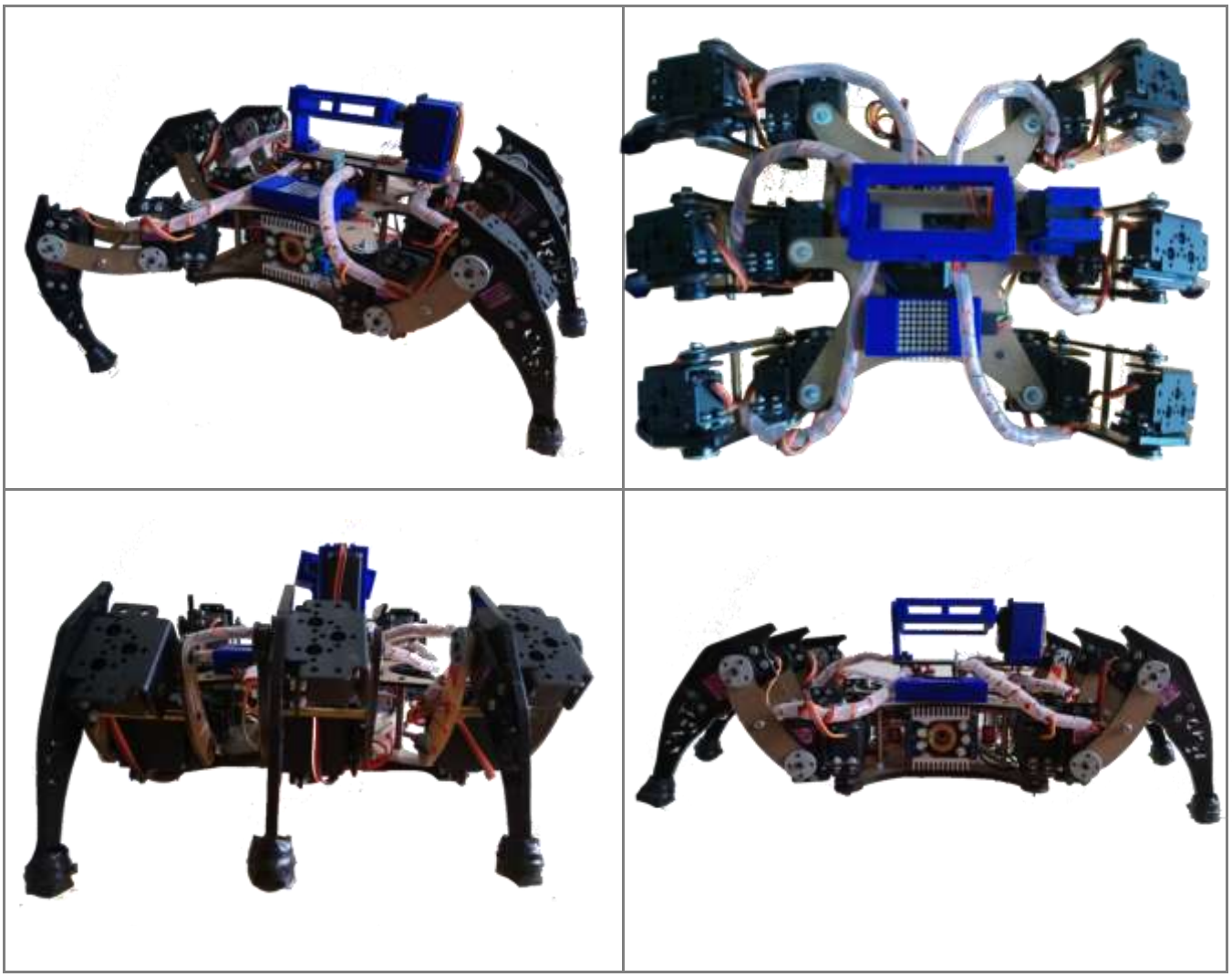
The final step is boundary limitation which we create a tracking area for checking to send controlling signal to make Robot keep track of the chosen object.

CHAPTER 5 THE EXPERIMENT RESULTS/FINDING AND ANALYSIS

5.1 PRACTICAL MODEL

Robot is able to respond to forward, back, left and right controls with a solidly designed structure, steady bearing legs, and a standing robot on a very balanced and stable surface.

Table 5. 1: The practical model.



5.2 RESULT'S SIMULATION MATLAB

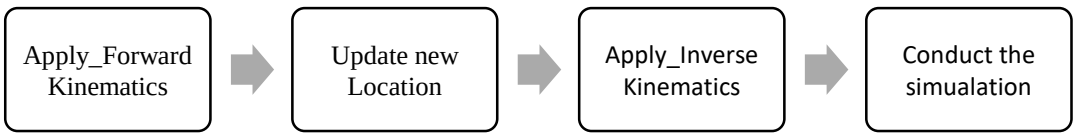


Figure 5. 1: The Matlab simulation process.

By using Matlab simulation, we experience the robot's movements to meet different moving targets, after applying the forward and inverse kinematics calculations into the fixed size robot model, we figured out the appropriate control coordinates for the program robot's movements based on 3 main calculations:

- Initialize the DH table:

```
%...
%initialize the DH table
D_H = [0, 0, z, the(1,1);...
       90, l1, 0, the(1,2);...
       0, l2, 0, the(1,3);...
       0, l3, 0, 0];
for i=1:3
    T_i_1_i = [cosd(D_H(i, 4)) -sind(D_H(i, 4)) 0 D_H(i, 2);...
              sind(D_H(i, 4))*cosd(D_H(i, 1)) cosd(D_H(i,
4))*cosd(D_H(i, 1))
              -sind(D_H(i, 1)) -sind(D_H(i, 1)*(D_H(i, 3)));...
              sind(D_H(i, 4))*sind(D_H(i, 1)) cosd(D_H(i,
4))*sind(D_H(i, 1))
              cosd(D_H(i, 1)) cosd(D_H(i, 1))*D_H(i, 3);...
                 0 0 0 1];
    TACC = TACC* T_i_1_i;
end
T_0_i = TACC;
T_end = T_0_i*[l3;0;0;1];
%...
```

- Apply Forward kinematics:

```
%...
%Forward Kinematics
for i=1:6
    if( i <= 3)
        P(i, 1) = cosd(the(i,1))*cosd(the(i,2))*l2+
       cosd(the(i,1))*l1 +…
       cosd(the(i,1))*cosd(the(i,2) + the(i,3))*l3;
    else
        P(i, 1) = -cosd(the(i,1))*cosd(the(i,2))*l2-
       cosd(the(i,1))*l1 +… -cosd(the(i,1))*cosd(the(i,2) +
       the(i,3))*l3;
    end
    P(i, 2) = sind(the(i,1))*cosd(the(i,2))*l2+
   sind(the(i,1))*l1 +… sind(the(i,1))*cosd(the(i,2) +
   the(i,3))*l3;
    P(i, 3) = sind(the(i,2))*l2 + sind(the(i,2) + the(i,3))*l3;
end
%...
```

- Apply Inverse kinematics:

```
%...
```

```
%Inverse kinematics
for i=1:6
    the(i,1) = atan2d(P(i,2), P(i,1));
    A = -l1 + P(i,1)*cosd(the(i,1)) + P(i,2)*sind(the(i,1));
    B = -P(i,3);
    cos3 = ((A^2 + B^2)-(l2^2 + l3^2))/(2*l2*l3);
    sin3 = sqrt(1 - cos3^2);
    the(i,3) =  - atan2d(sin3, cos3);
    C = B*(l3*cos3 + l2) - l3*sqrt(1 - cos3^2)*A;
    D = A*(l3*cos3 + l2) + l3*sqrt(1 - cos3^2)*B;
    the(i,2) =  - atan2d(C, D);
End
%...
```

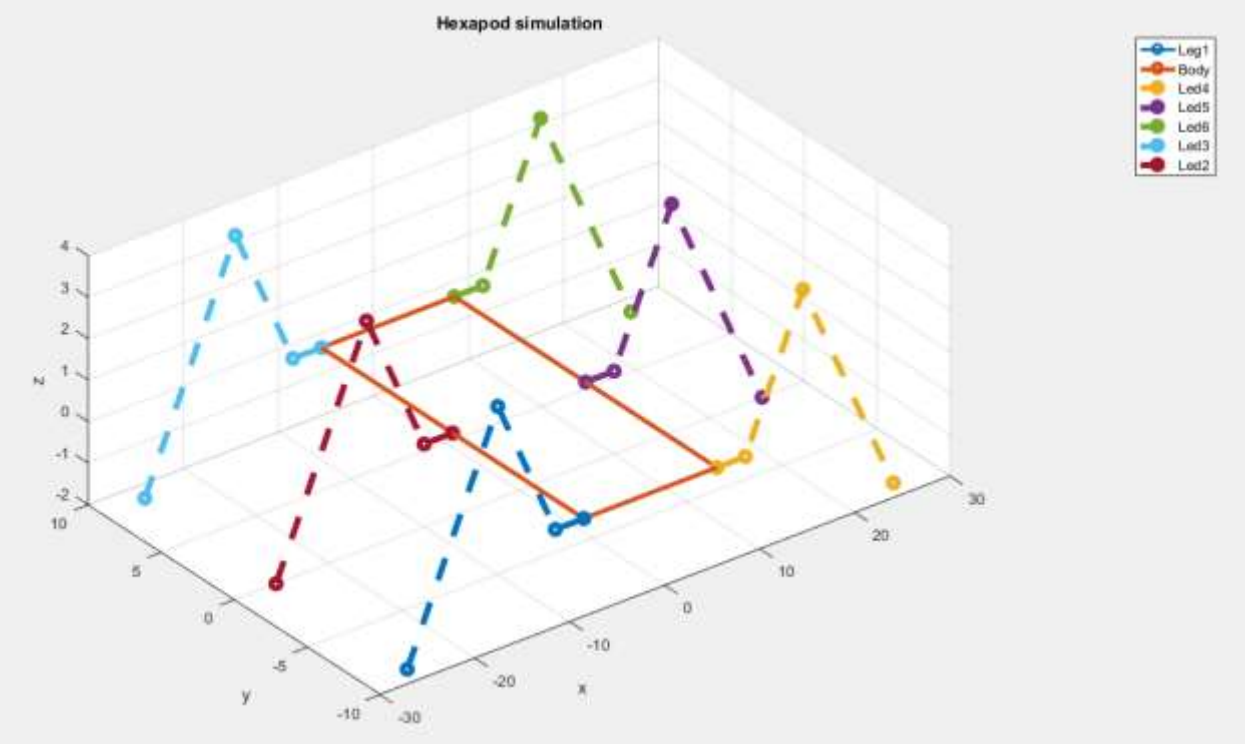
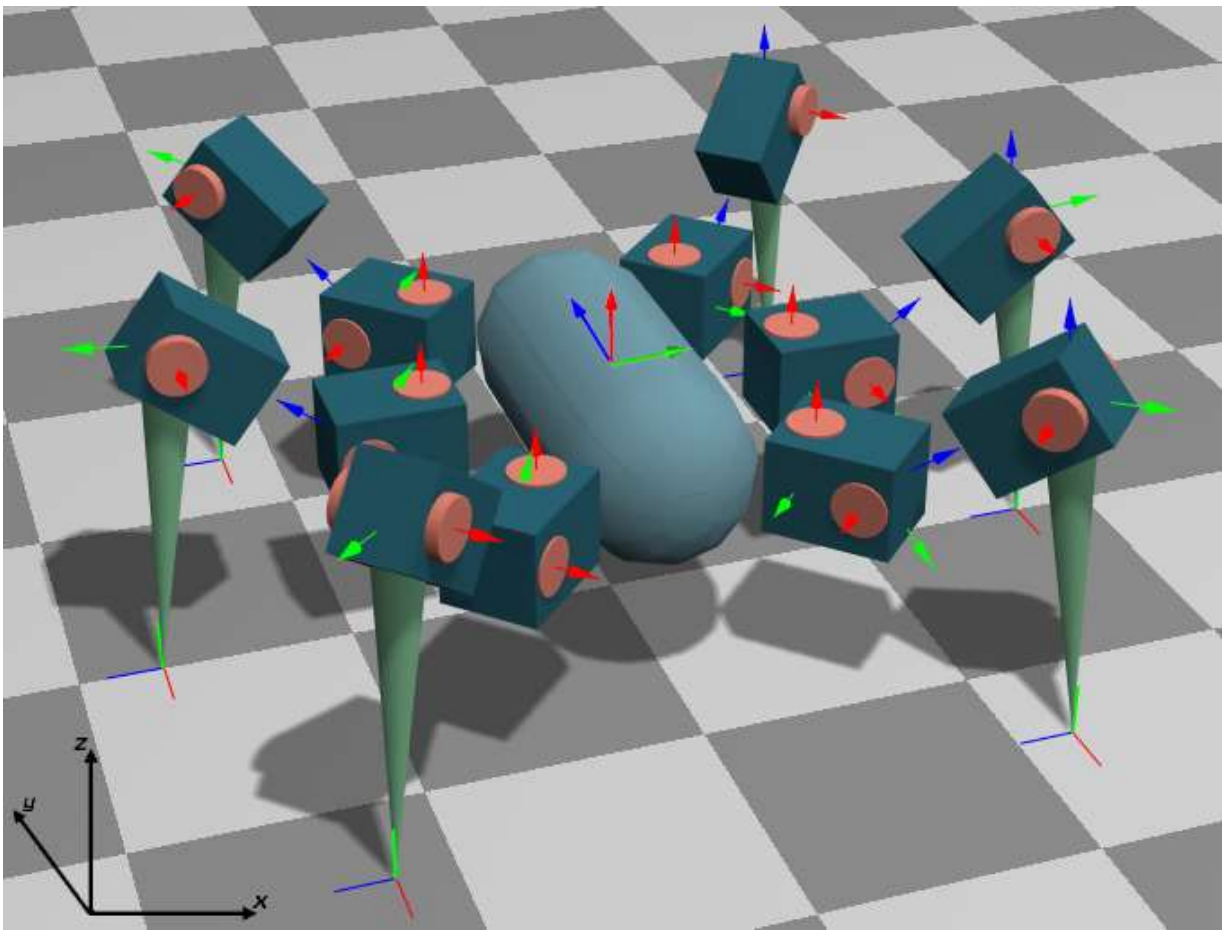
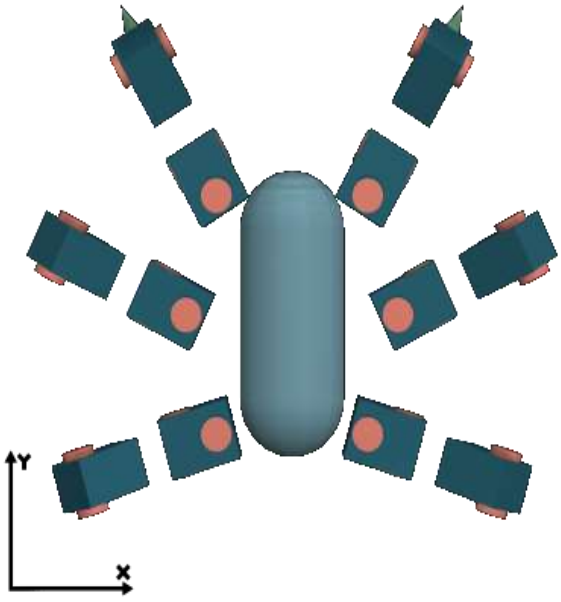
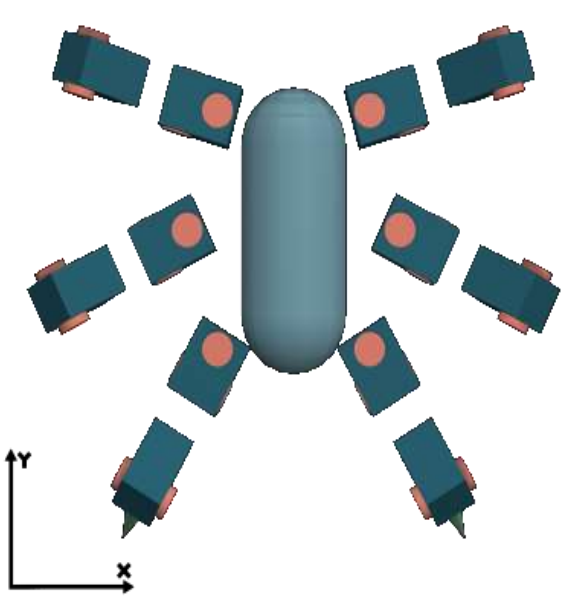
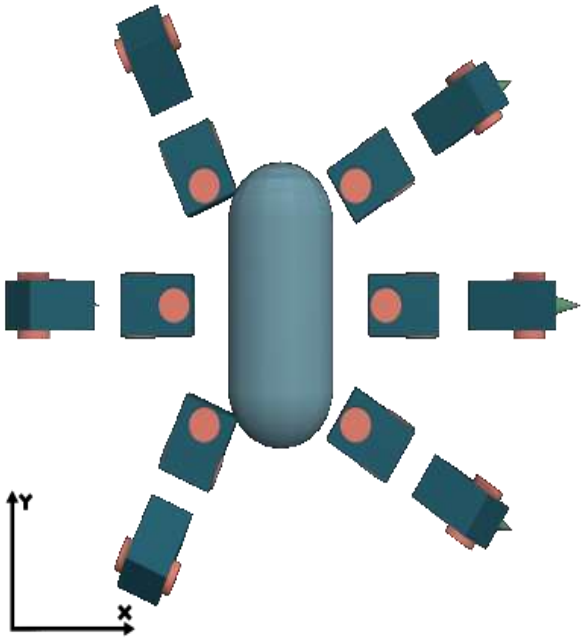
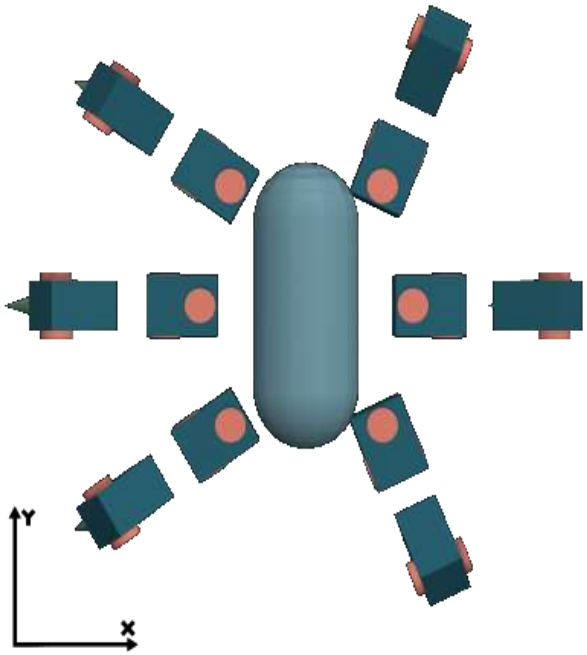
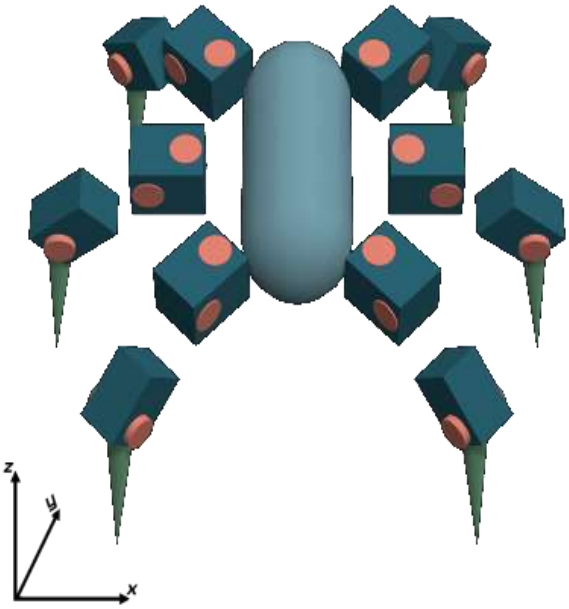
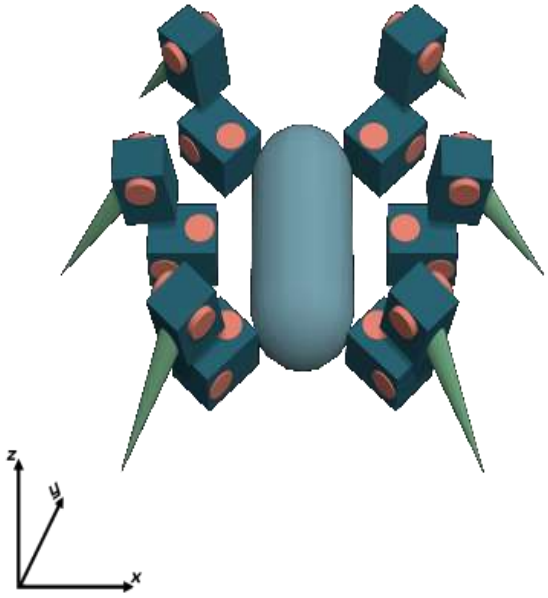


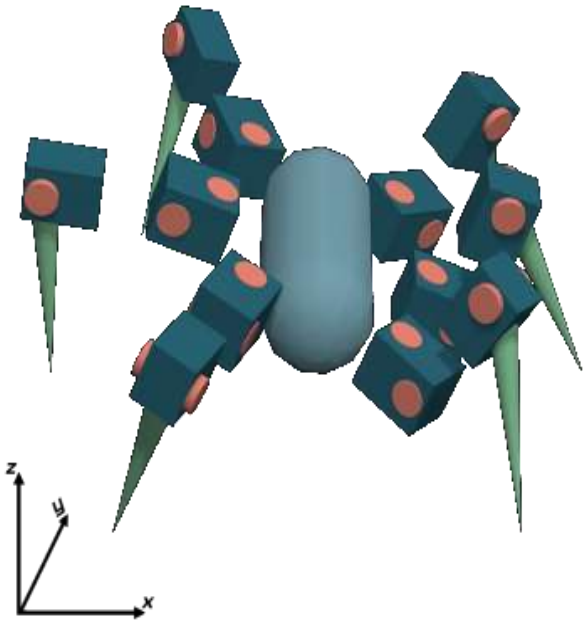
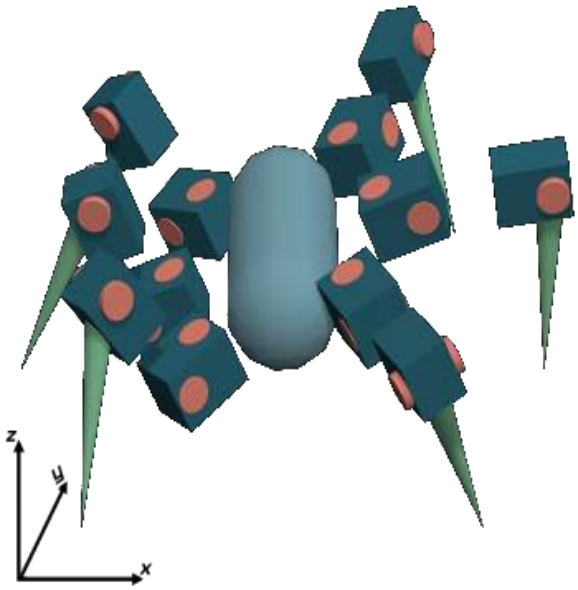
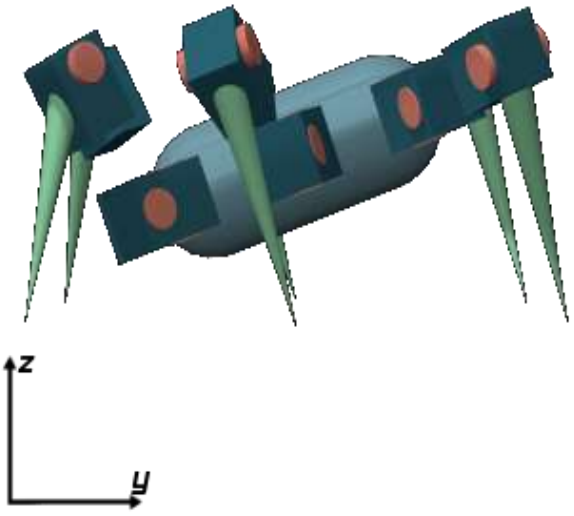
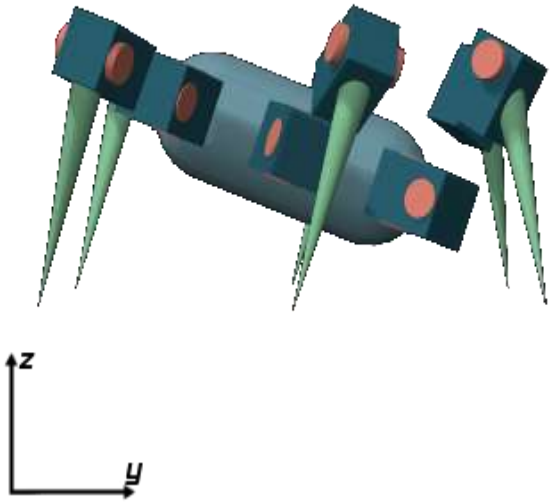
Figure 5. 2: The hexapod simulation in matlab.

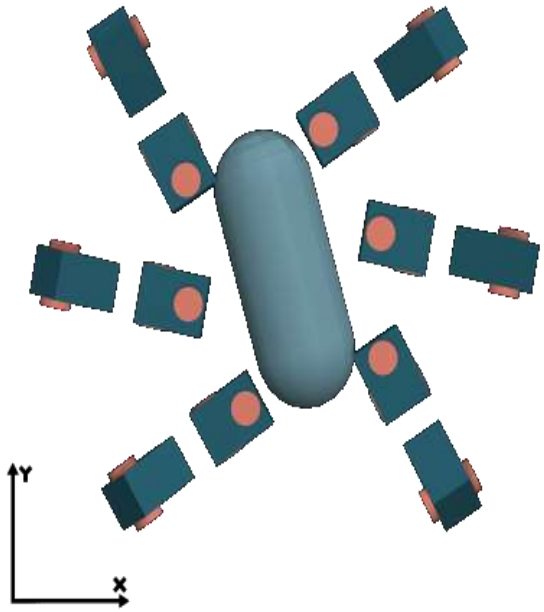
By using an open-source software called InsectRobotSimulation released by User Kris Temmerman on *Github.com*, we could simulate the basic movement patterns of the Robot. There are the basis for building Hexapod's movements. (Roll - pitch - yall… etc)



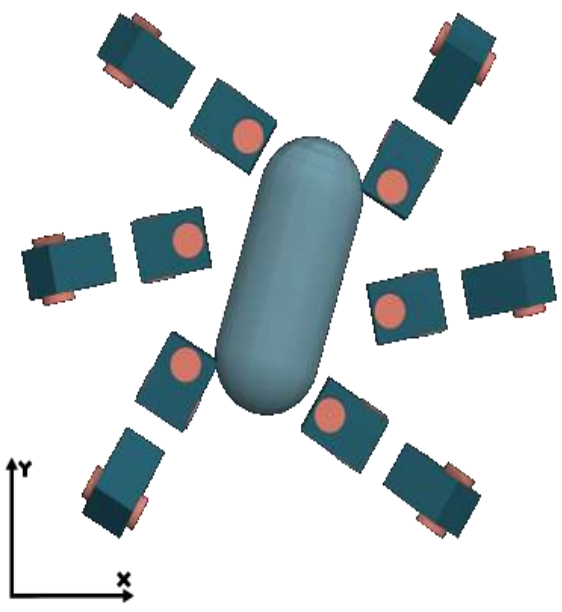
	
Body Offset - Negative direction	Body Offset - Positive direction

	
Body Offset - Negative sideways	Body Offset - Positive sideways
	
Body Height - Up	Body Height - Down

	
Body rotation - Counter clockwise Roll	Body rotation - Clockwise Roll
	
Body rotation - Clockwise Pitch	Body rotation - Counter clockwise Pitch



Body rotation – C.Clockwise Yall



Body rotation - Clockwise Yall

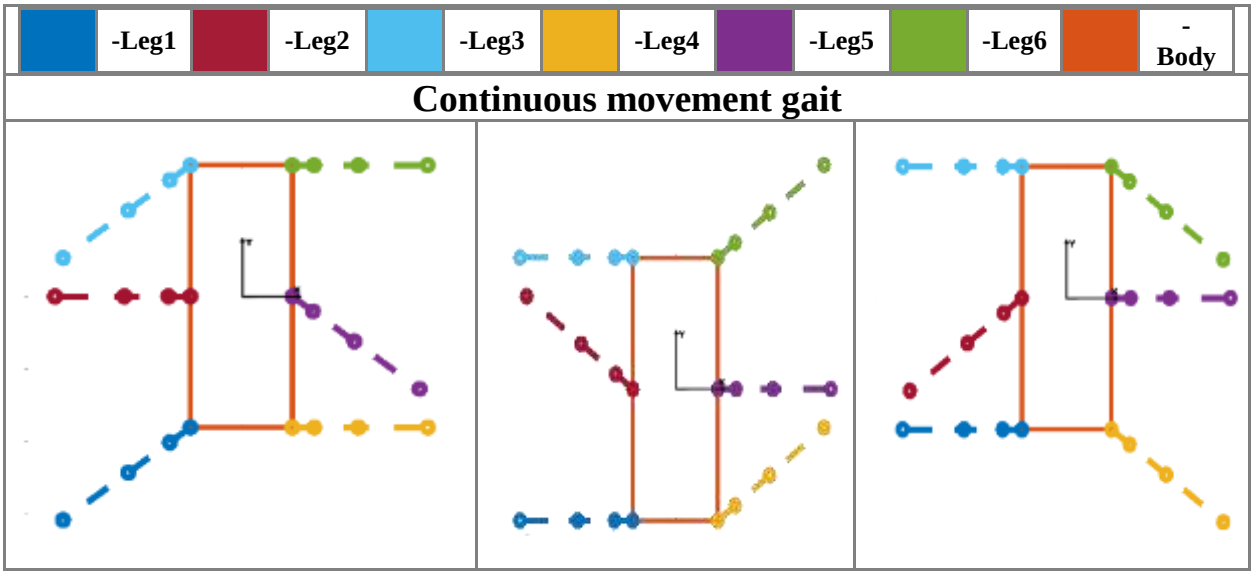
Using Matlab simulations, we can experience the optimal movements for robots. After trying to combine the basic movements, the robot established 3 gaits in accordance with the requirements of designing Robot.

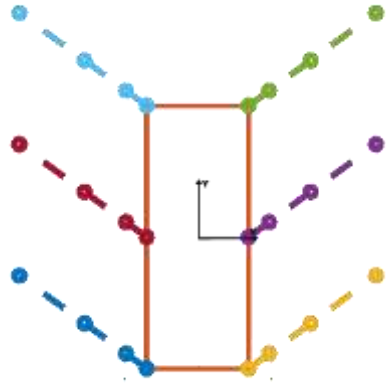
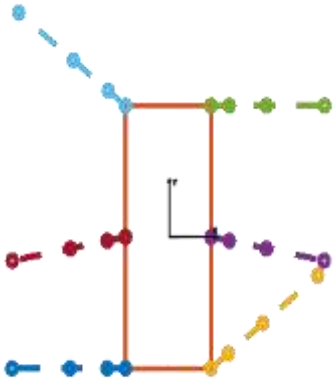
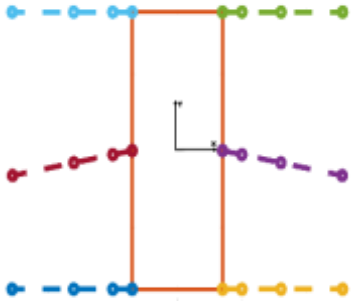
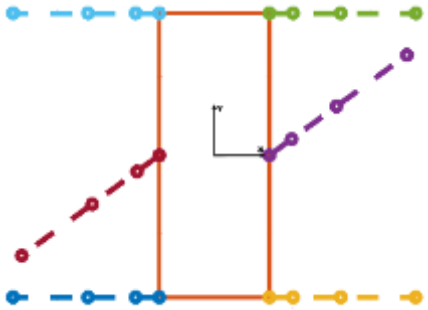
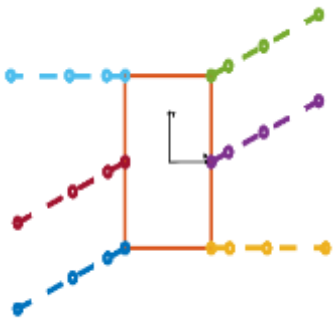
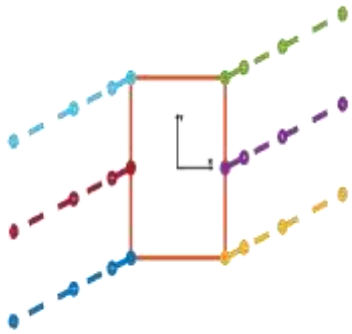
1. Continuous movement gait

2. Leaning-over movement gait

3. Turning gait

Consists of 4 steps to complete the movement of one leg: Hang the leg - Move to the position - Put the leg down - Pull the body back.



1, 3, 5 move	1, 3, 5: reset, 2, 4, 6: move	2, 4, 6: reset and repeat
Leaning-over movement gait		
		
1 - 6: move	2, 5: reset over	1, 3, 4, 6 reset and repeat
Turning gait		
		
2, 5 move	1, 6 move	3, 4 move and pull body back

5.3 KEEP BALANCING RESULT

- Set-point: 90 degree
- Sample time: 50ms

The response of the robot is relatively good with the plane as well as the inclined terrain when put into balance mode.

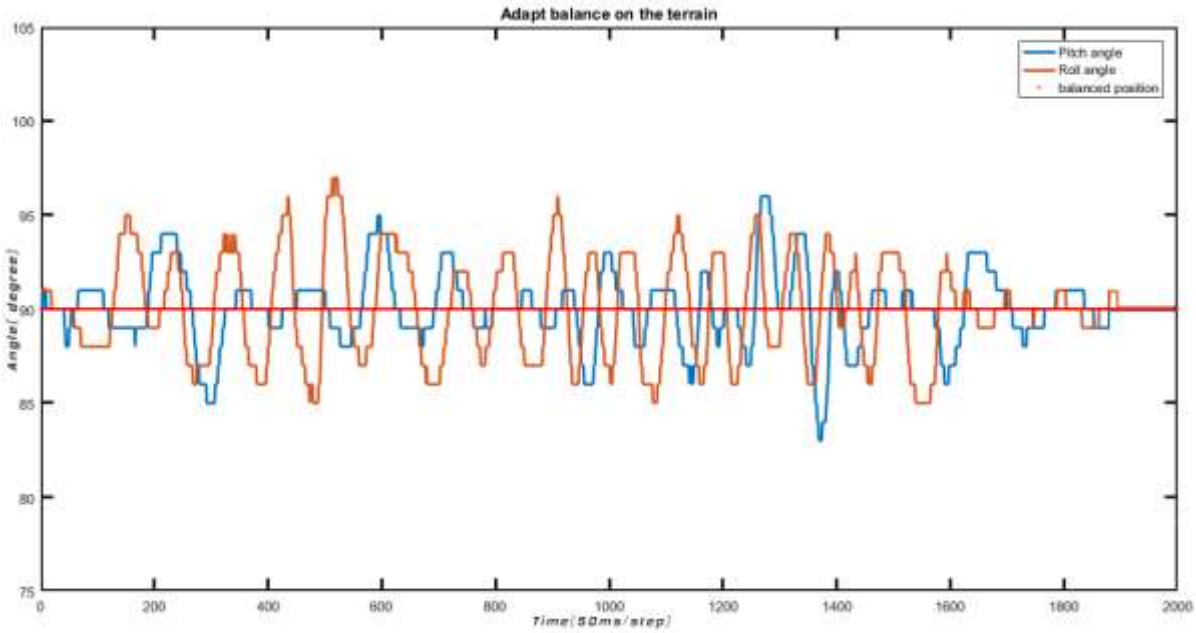


Figure 5. 3: Adapt balance on the terrain.

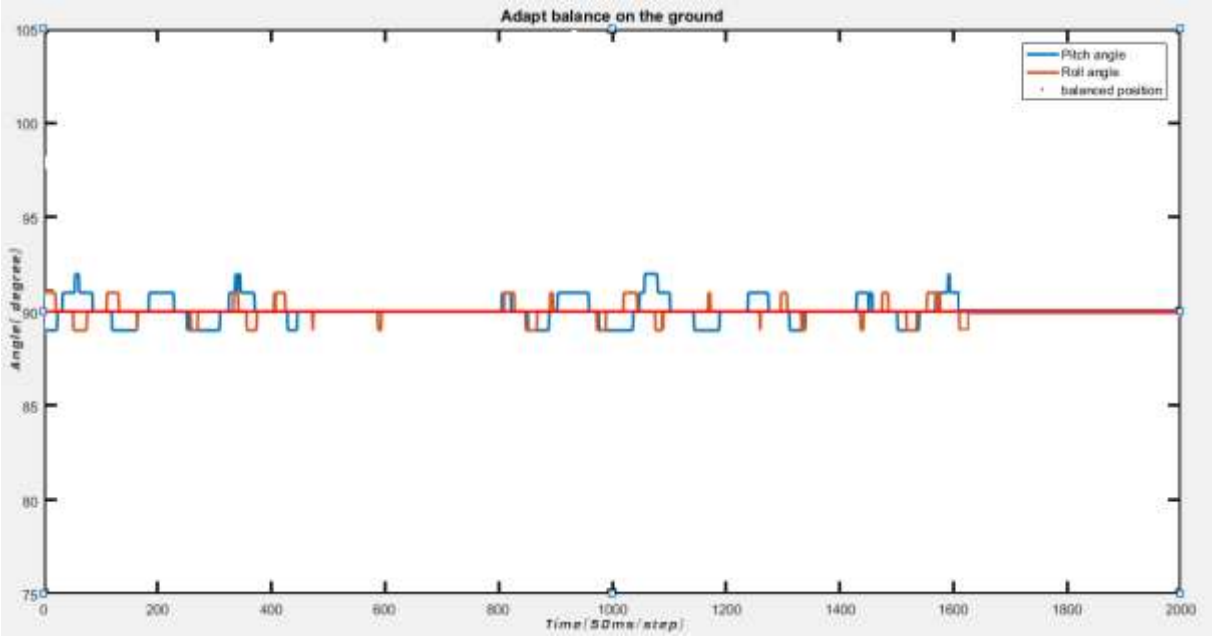
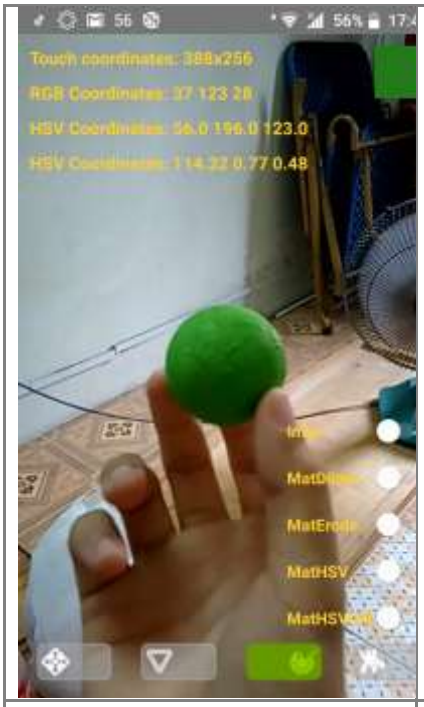
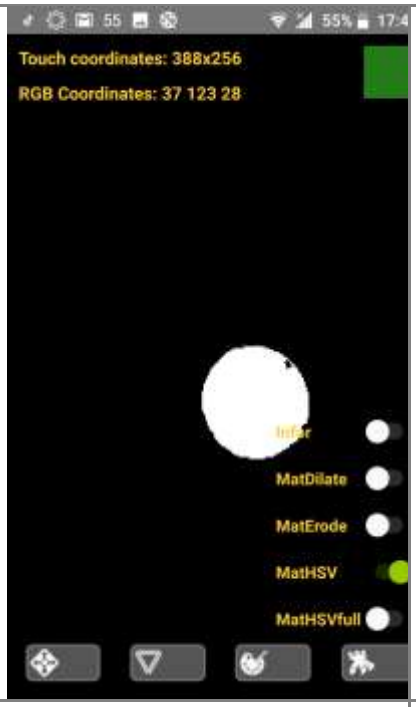
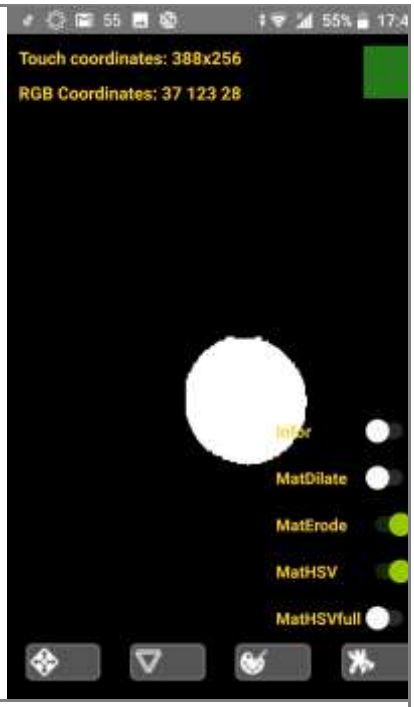
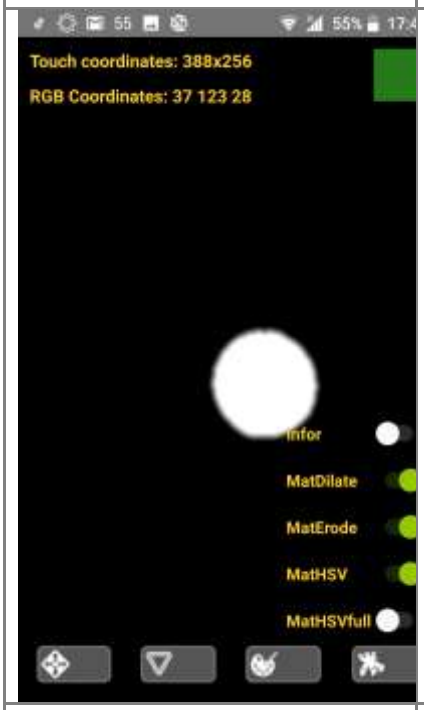
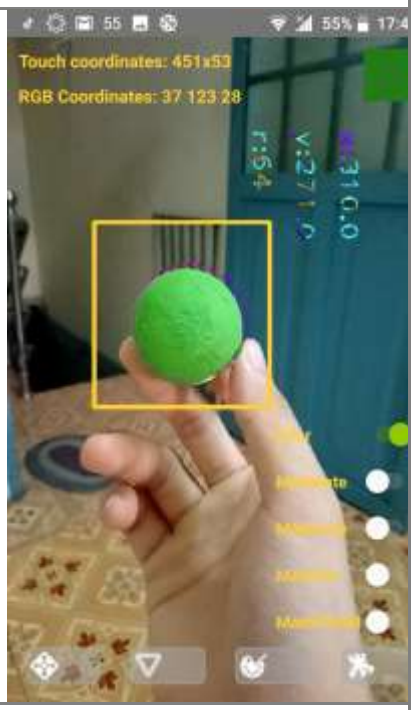


Figure 5. 4: Adapt balance on the ground.

5.4 IMAGE PROCESSING AND TRACKING RESULTS

Image processing and object tracking work stably, separating the color part and detecting round shape object.

Table 5. 2: The results of the image processing.

		
In put frame with chosen-colour value in RGB and HSV space.	Split-ted area has fitted with HSV colour range chosen by user.	Apply Morphological Transformations.
		
Apply GaussianBlur to reduce noise.	Apply Hough-circle transform to find the round object with its attribution.	Keep tracking object in allowed area.

5.5 EXPERIMENTAL RESULT

5.5.1 MANUAL MODE

Table 5. 3: The parameter table evaluating movement.

	Experiment times	Movement error under controlling	Efficiency
Forward	20	3.5%	96.5%
Backward	20	3.5%	96.5%
Turn-left	20	2.34%	97.7%
Turn-right	20	2.34%	97.7%
Balancing	20	1.5/90 ~ 1.67%	98.3%

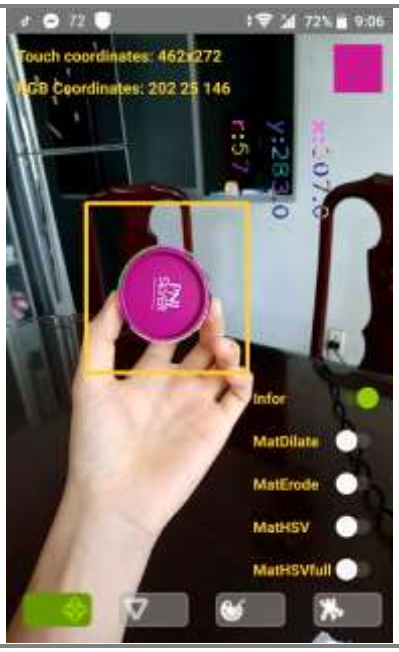
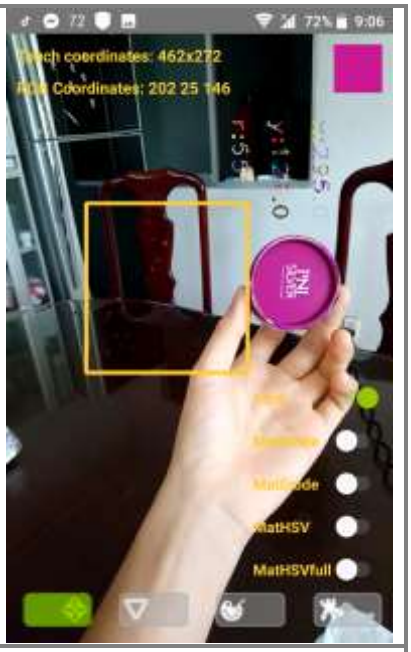
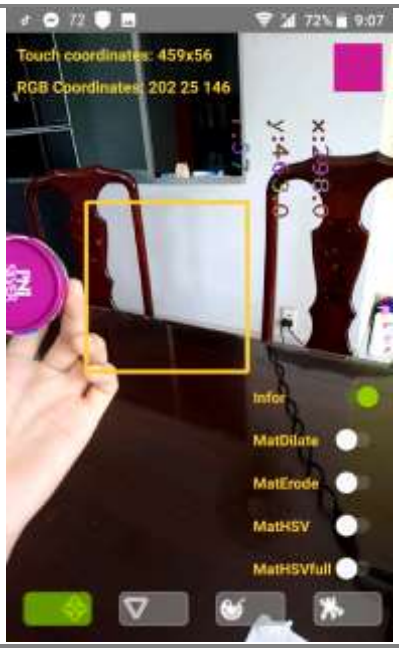
5.5.2 TRACKING MODE

Table 5. 4: The parameter table evaluating the accuracy of tracking.

	Experiment times	Response time	Efficiency
Forward	20	1.5 - 3s	99%

Figure 5. 5: Keep tracking object in allowed area.

Table 5. 5: Define the parameters of the object.

		
- Original image before object tracking.	- Touch coordinates 462*272 px - RGB Coordinates: 202 - 25 - 146 - Centre coordinates: 207 - 283 px - Radius: 57 px - Centerchecked();	- Touch coordinates 462*272 px - RGB Coordinates: 202 - 25 - 146 - Centre coordinates: 295 - 121 px - Radius: 55 px - Turnright();
		
- Touch coordinates 462*272 px - RGB Coordinates: 202 - 25 - 146 - Centre coordinates: 457 - 277 px - Radius: 59 px - Downward();	- Touch coordinates 462*272 px - RGB Coordinates: 202 - 25 - 146 - Centre coordinates: 298 - 463 px - Radius: 57 px - Turnleft();	- Touch coordinates 462*272 px - RGB Coordinates: 202 - 25 - 146 - Centre coordinates: 145- 277 px - Radius: 56 px - Upward();

```
private static int res_XSony = 1080, res_YSony = 1920;
private static final int res_Xmetric = 1700, res_Ymetric = 1080;
private static final int res_X = 720, res_Y = 480;
private static final int minx = 200, maxx = 400, miny = 180, maxy = 380, rmin = 25, rmax = 40; //200 400 180 380
```

When the smart-phone tracks the object, the coordinates of the center will be used to compare with the preset coordinates of the center on the screen (yellow square pattern), when the object is off-center, the program will execute the corresponding functions to send the signals to Micro-controller which is used for the robot to control and follow the selected object:

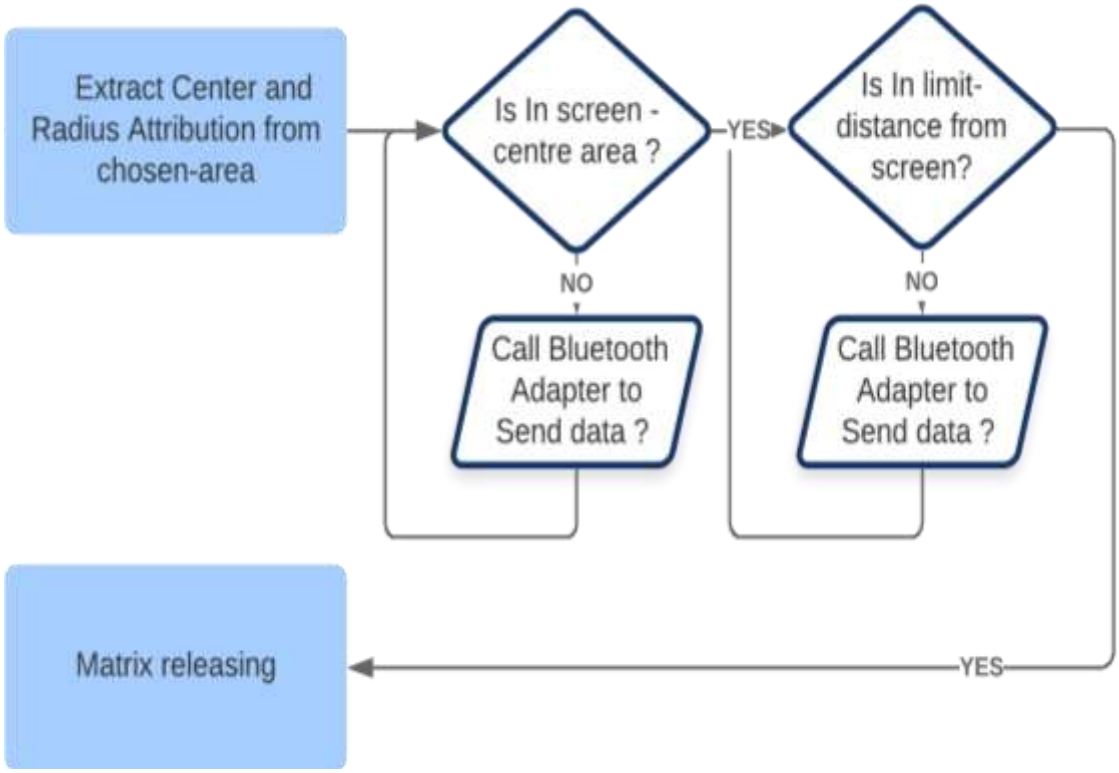


Figure 5. 6: The image processing and data transmission.

By taking out the radius of the object to be tracked, the robot can detect when objects are either near or far away so that it can follow to maintain distance.

5.6 DATA TRANSMISSION RESULT

Data transmission between device and Google-firebase in real time is stable, the data is updated on the chart and stored in Firebase user database.

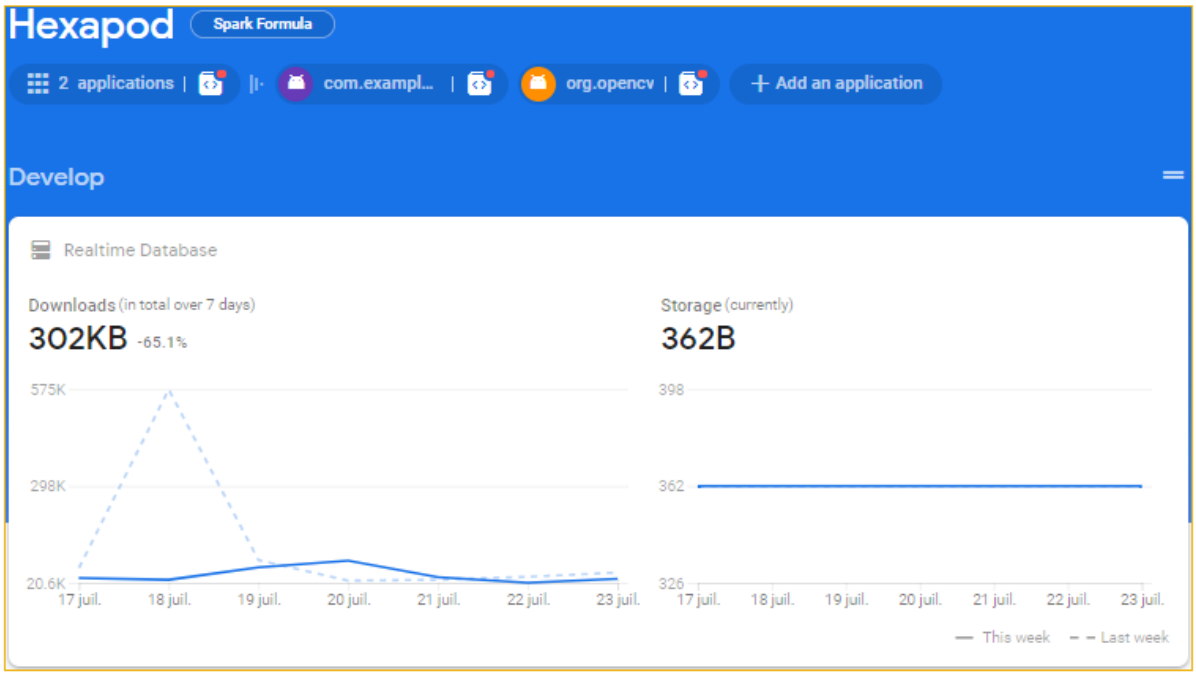


Figure 5. 7: The data transfer rate.

Figure 5. 8: Monitor parameters online.

Figure 5. 9: Far away mode.

In the far away mode, we applied firebase technology to transmit data through the internet.

The 1st phone and 2nd phone linked together using the google firebase. The task of the first phone is to transmit data to the second phone.

The task of the second phone is to receive data from the first phone and connect Bluetooth of the hexapod.

- Describe the operation process:

-The 1st screen displays the robot's movement feature. Example: forward, backward, turn left, turn right, etc.

-The 2nd screen displays the data received from firebase.

-The signal control transmits from 1st phone to 2nd phone and control hexapod by Bluetooth.

CHAPTER 6 CONCLUSION AND RECOMMENDATIONS

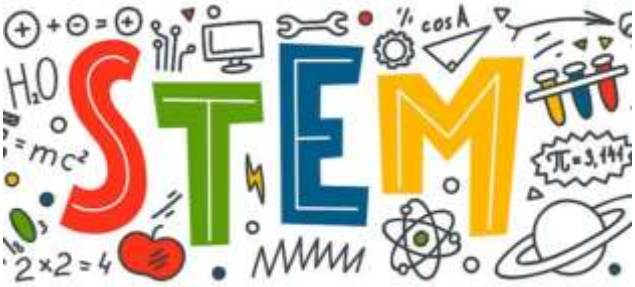
6.1. CONCLUSION:

After 5 months of research and development together, our team has completed the thesis "Design, Implementation and Control of Hexapod Robot combining image processing on Android platform" with the set goals achieved:

- The robot is compact in size, flexible in operation.
- Robot works effectively and achieves the required standard.
- Transfer data by communicating to smartphone via Bluetooth platform.
- Transfer data in real-time database via Firebase for long distance controlling.
- Robot can track the object based on the setting conditions.
- Read angle correctly by using Kalman filter.
- Robot can balance, move forward, backward, turn left, turn right on even terrain.
- The android software works well and is absolutely easy to approach.

Besides that, this project also has some limitations such as Robot's motion limit angles depends on the hardware so Robot will have some blind spots, the size and mass of robots is relatively large, leading to moving not so fast.

6.2. RECOMMENDATIONS:



Based on the successful research of the thesis “Design, Implementation and Control of Hexapod Robot combining image processing on Android platform”, We can improve the robot's operability and power with better equipment, lighter materials such as carbon fiber, both increasing durability and helping robots operate in different harsh environments. Applications of Hexapod in transporting, rescuing and executing search missions are feasibly appropriate. In addition, the flexibility of the Robot is also improved by applying many different algorithms to build methods of moving on many terrain types included hills, swamps, sandy deserts, and rocky soil…etc.

By the way, through the construction of robot control software on smart-phone that integrates the functional stages. We look forward to building an STEM education ecosystem, which brings an objective view of Robots and 4.0 technology in general to everyone. By easily accessing HEXAPOD software on OS platforms that we research and develop, with its lively design interface and easily interactive feature, the project is possible for anyone. People can own a robot and study how it works. That is the biggest goal we want to achieve in this project.

REFERENCE

English

[1] John J.Craig, *Introduction to Robotics*, Third. United States of America: Pearson Prentice Hall, 2005.

[3] Zihao Zang, "Mechanism Design and Kinematics Analysis of Spider -like Octopod Robot," *IOP Publ.*, 2019.

[4] Rotation Matrix, "https://en.wikipedia.org/wiki/Rotation_matrix."

[5] G.Saravanan and G.Yamuna, "Real Time Implementation of RGB to HSV/HSI/HSL and Its Reverse Color Space Models," *Internaltional Conf. Commun. Signal Process.*, vol. 978, no. 1, 2016.

[6] Joy Christy A and U. A, "A Novel Percentage Split Distribution Method for Image Thresholding," *Pre-proof*, 2020.

[7] Mathematical morphology, "https://en.wikipedia.org/wiki/Mathematical_morphology."

[8] Gaussian Blur, "https://en.wikipedia.org/wiki/Gaussian_blur."

[9] S. Marschner and A. Efros, *Introduction to Visual Cmputing*. 2019.

[10] Canny Edge Dectection, "https://en.wikipedia.org/wiki/Canny_edge_detector#Gaussian_filter," 2019.

[11] O. R. Vincent, "A Descriptive Algorithm for Sobel Image Edge Detection A Descriptive Algorithm for Sobel Image Edge Detection Clausthal University of Department of Computer," no. January 2009, 2014.

[13] J.Canny, *A Computational Approach to Edge Detection*. IEEE, Pattern Anal. And Mach, Intell.

[14] Prajwal Shetty, "Circle Detection in Image," pp. 1–30, 2011.

[15] Virendra Kumar Yadv and etc, "ApproachtoAccurateCircleDetectionCircular." IEEE, India, 2014.

[16] M. Z. Zhang and H. R. Cao, "A New Method of Circle's Center and Radius Detection in Image Processing." IEEE, Automation and Logistics, 2008.

[17] Ramsey Faragher, "Understanding The Basic of the Kalman Filter Via a Simple and Intuitive Derivation." IEEE Signal Processing Magazine, pp. 128–132, 2012.

[18] How a Kalman filter works, "https://www.bzarg.com/p/how-a-kalman-filter-works-in-ictures/?fbclid=IwAR1miULah3GEVi8rP140lmjZFi1t4kJECsjM8jQLRDTVAHCGLipX7Bh9M-A," *Mini-courses*, 2015.

Tiếng Việt

[2] Ph.D. Nguyen Van Thai - Robotics, "https://bit.ly/2PWurHM" , 2017.

[12] Nguyễn Văn Long, *Ứng dụng xử lí ảnh trong thực tế với thư viện OpenCV C/C++*. 2017.


TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH
www.hcmute.edu.vn