# - Introduction

## Unmanned Aerial Vehicles (UAVs)

## The Miniature Quad-rotor Unmanned Aerial Vehicle

## Anatomy of the Quad-rotor Helicopter

### Frame

### Landing Gear

### Motors and Propellers

### Battery

### Sensors

### Flight Control Board

### Transmitter and Receiver

## Basic concepts of the quad-rotor helicopter

### Throttle

### Roll

### Pitch

### Yaw

## Applications of Miniature Quad-rotor Helicopters

### Border Patrol

### Disaster Management/ Search and Rescue

### Wild fire detection

### Photography

### Military and Law enforcement

### Research

### Agricultural and Industrial applications

## Chapter Summary

# - Literature Review

## Previous works on the quad-rotor helicopter

## The basic concepts of Artificial Neural Networks

### Applications of Artificial Neural Networks in Aircraft Control

## The basics concepts of PID controller

### Applications of PID controller

## Aim and Objectives of the research project

## Control Tuning Technique

### Ziegler Nicolas method

### Trial and error

## Contributions of this work (to be finalised)

## Thesis layout

## Chapter Summary

# - Working Principles and Analytical Dynamic Model of Quad-copter

## The Newton-Euler model

### Coordinate Frames

### Quad-rotor Modelling Assumptions

### Quad-rotor Helicopter State Variable definition

### Direction Cosine Matrix

### Quad-rotor Kinematics

### Quad-rotor Dynamics

### Quad-rotor Aerodynamic Forces

### Quad-rotor Moments (Torques)

### Quad-rotor Moments of Inertia

### Equations of Motion

## Actuator Dynamics (DC-motor)

### Voltage and Angular Velocity of Propeller

### Voltage and Thrust

### Rolling Moment

### Pitching Moment

### Yawing Moment

### Acceleration along the x-axis

### Acceleration along the y-axis

### Acceleration along z-axis

## Chapter Summary

# - Simulation of Quadcopter Model in Matlab/Simulink and 3D animation

## Matlab/Simulink Software

## Model Implementation in Matlab/Simulink

### Summary of equations of motion

### Actuator Subsystem

### Roll Subsystem

### Pitch Subsystem

### Yaw Subsystem

### X-Motion Subsystem

### Y-Motion Subsystem

### Z-Motion Subsystem

## Running the Simulation

### Calibration and Preliminary Calculations

### Hover

### Throttle (Vertical Motion)

### Roll

### Pitch

### Yaw

## 3D animations (in progress)

### 3D Quadcopter model

### Euler rotation

### Quaternion

### Animation Results

## Chapter Summary

# - Publications

1. *Abdelkader Fareha; Amar Bousbaine; Ajay K. Josaph* “An Integration of 6DOF Quadcopter MATLAB/Simulink Controller Algorithm onto a PIXHAWK Autopilot”, The 10th International Conference on Power Electronics, Machines and Drives, PEMD, 15 - 17 December 2020 | Online Conference.
2. Emmanuel Okyere1, Amar Bousbaine, Gwangtim T. Poyi, Ajay K. Joseph, Jose M. Andrade*,” LQR controller design for quad-rotor helicopters” The Journal of Engineering,* ISSN 2051-3305, doi: 10.1049/joe.2018.8126 , pp4003-4007, 17th June 2019.
3. Bousbaine, A. Bamgbose, G.T. Poyi and A. K. Joseph "Design of Self-tuning PID Controller Parameters Using Fuzzy Logic Controller for Quad-rotor Helicopter" Published in International Journal of Trend in Research and Development (IJTRD), ISSN: 2394-9333, Vol. 3, Issue-6 , December 2016.
4. *Ajay K Joseph; Amar Bousbaine; Abdelkader Fareha, “A Wireless communication system for a quadrotor helicopter”, 2018 53rd International Universities Power Engineering Conference (UPEC),* 4-7 Sept.  *2018, Glasgow* **DOI:**[10.1109/UPEC.2018.8542040](https://doi.org/10.1109/UPEC.2018.8542040).
5. *Abdelkader Fareha; Amar Bousbaine; Ajay K. Josaph, “ Experimental Characterisation of quad rotor controller based on Kalman Filter”, 53rd International Universities Power Engineering Conference (UPEC),* 4-7 Sept. 2018, Glasgow, **DOI:**[10.1109/UPEC.2018.8541858](https://doi.org/10.1109/UPEC.2018.8541858).

# Work to be completed

Ø  Abstract

Ø  Table of content

Ø  List of figures

Ø  List of tables

Ø  Nomenclature

# – Controller Design Methodology (Kalman Filter and PID) (A rough content of the chapter)

## Linearization and State Space Representation for system

### Vertical system

### Directional

### Latitudinal

### Longitudinal

### Controllability and Observability of the systems

## Flight control Algorithm

### Control Technique

## Kalman Filter Algorithm

### Altitude

### Directional

### Latitudinal

### Longitudinal

## Experimental Identification of the physical Parameters

### Moment of Inertia – Bifilar pendulum

### Motor Torque – Load cell

### Parameters’ extraction procedure

### Parameters extraction

## Software implementation and Simulation Results

### Model Implementation

### Pre-existing Model

### Flight controller modelling

### Kalman filter modelling

#### Trajectory generator model

#### Noise Generator

#### Quadcopter Mixer

#### PWM Scaling

### Simulation Results

## Summary

# Sensor Fusion and Wireless Communication Systems

## Different types of Sensors and Communication systems

### Sensors

#### Ultrasonic

#### IMU

#### Camera

### Communication Systems

#### UART

#### I2C

#### Bluetooth (HC-05)

#### TCP/UDP

## Arduino (incorporate sensors)

### Bluetooth (HC-05)

### Ultrasonic modules

### IMU 6050

### I2C programming

### UART communication

### CRC for error correction

### Result of sensor fusion

## Raspberry Pi 3 (RPI 3)

### Ultrasonic sensors

### I2C in RPI

### UART in RPI

### UDP Protocol

### Camera module

### Results of sensor fusion

# - Conclusions and further work

# - Two papers

# - Equations

==================================================================================

## Equations

|  |  |  |
| --- | --- | --- |
|  |  | 1‑1 |

This system as shown in the equation *2*‑*1*

## 2D Rotation

A rotation is the process of changing an angle of an object/point. A point in a 2D space can be expressed in several ways. The two most common methods to express 2D point are Cartesian and Polar forms.

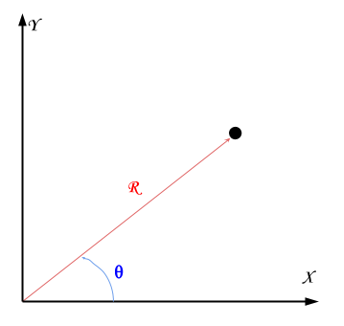
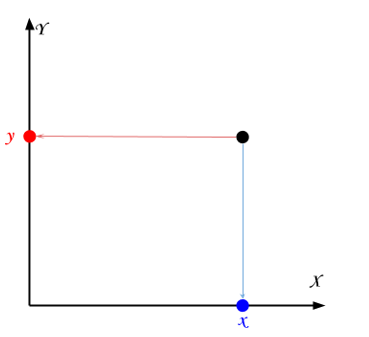


Figure 1‑1

Figure 1‑2

According to Euclidean form of orthogonal axes Figure 1‑1 represents distance measured to the point **A** in Cartesian method. Figure 1‑2 shows the distance from centre (axis of rotation) and the angle to the point A in polar method. Regardless two coordinates are required to represent a point in 2D.

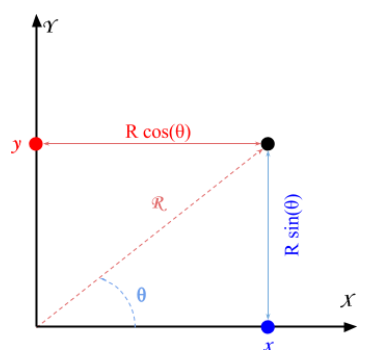


Figure 1‑3

The two methods of rotational representation mentioned above is superimposed in Figure 1‑3. Trigonometric equations are used to determine the length of its (point’s) projections onto the X and Y axes.

|  |  |  |
| --- | --- | --- |
|  |  | 1‑2 |
|  |  | 1‑3 |

Imagine the point (x, y) is rotated by an angle ф

<https://www.alanzucconi.com/2016/02/03/2d-rotations/>

## Figure

|  |  |  |
| --- | --- | --- |
|  |  | 2‑1 |

The equation *2*‑*1*

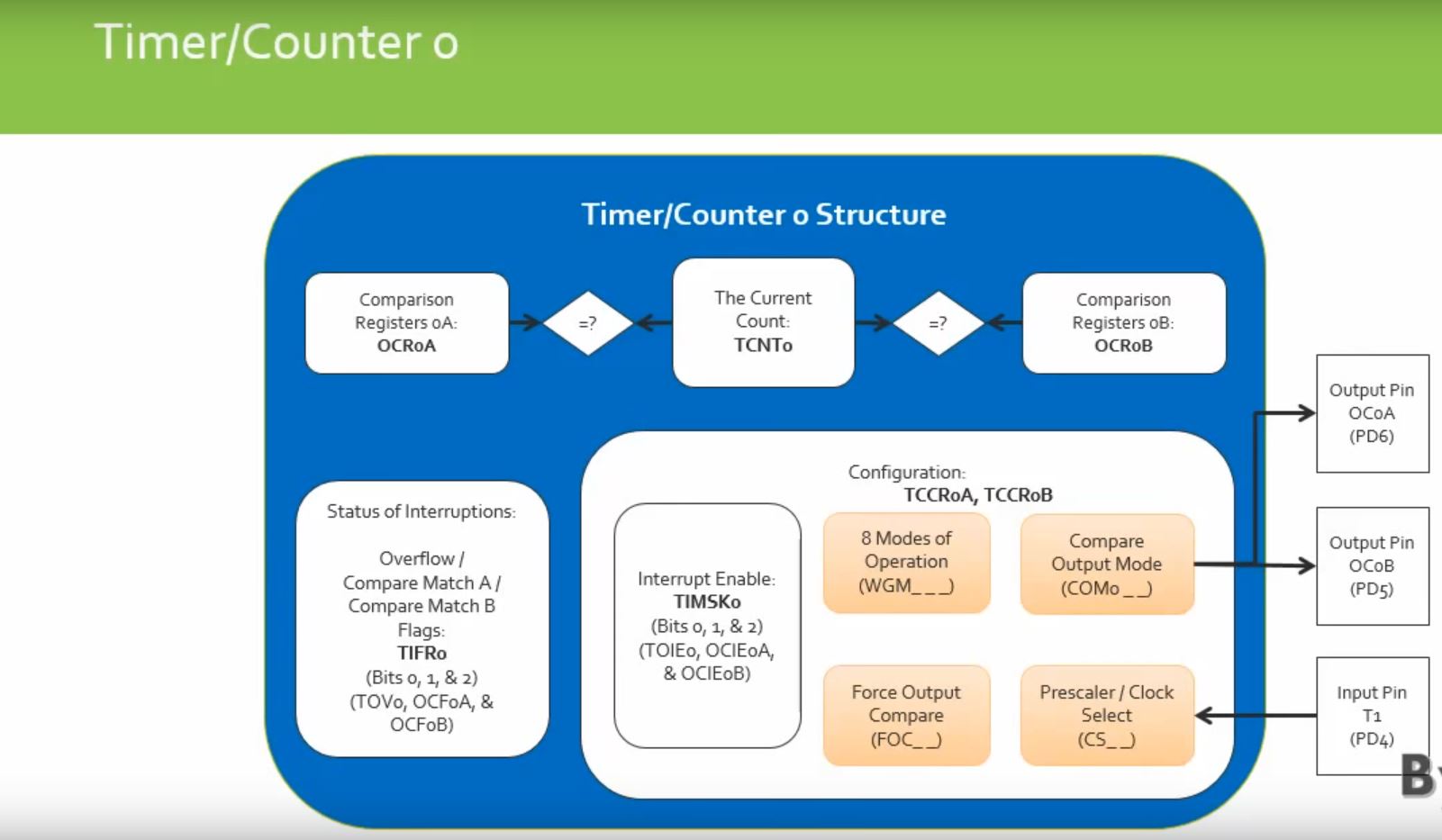


Figure 2‑1

The Figure 2‑1

## Tables

Table 3‑1

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

## Describes the working principle of I2C using MB1242 ultrasonic sensor

This chapter describes the fundamental of i2c. Also, describes how to debug I2C devices using a logic analyser.

### Description

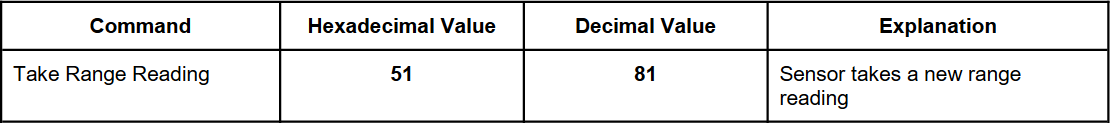
I2c is blah blah blah

### Principle

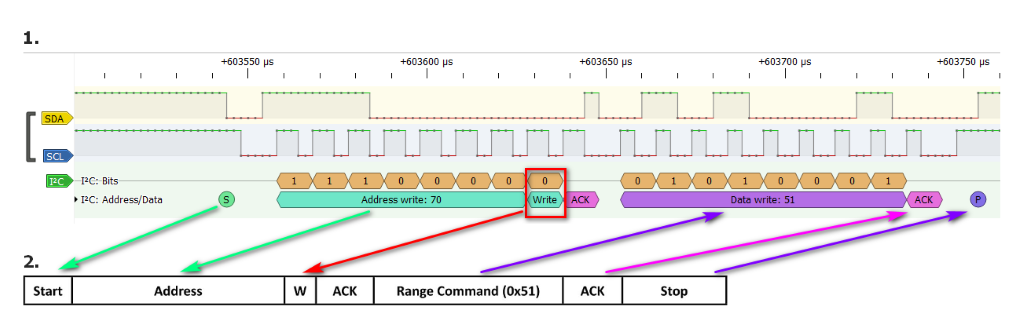
There could be a numerous amount of slave device on the i2c line. In order to communicate with an I2C device and to its internal features, there are two important things which are necessary to know before communicating with any slave devices.

1. Slave devices address
2. Whether to perform a **READ** or **WRITE** command

The datasheet of an I2C device will contain this information. The important factor is to understand the purpose of the slave device is whether to execute **READ** or **WRITE** sequence. In this scenario, the slave device **MB1242** ultrasonic sensor is required to have a **WRITE** at the end of the slave address (‘**0x70**’), followed by the pointer register requesting rang reading. Note the binary representation of the slave address contains ‘**0**’ at the least significant bit which represents a **WRITE** command. Whereas ‘**1**’ represents a **READ** command. The pointer register is not available on the datasheet for this particular device. However, register which is required to obtain the range reading is given as ‘**0x51**’ as shown below.



The initial sequence of operation carried out, as per datasheet, using i2c protocol for the ultrasonic module is illustrated in figure below.



The timing diagram shown above was obtained by a logic analyser and by running the Arduino code shown below.

1. #include "Wire.h"
3. **void** setup()
4. {
5. Serial.begin(9600);           // Enabled for printing and debuggin
6. Wire.begin();                 // Essential for I2C communication
7. }
9. **void** loop()
10. {
11. Wire.beginTransmission(0x70); // Slave address
12. Wire.write(0x51);             // Register to receive range
13. Wire.endTransmission();       // Ends I2C Transmission
15. delay(100);
17. Wire.requestFrom(0x70, byte(2)); //Sends a READ request of 2 bytes
19. **if**(Wire.available() >= 2)
20. {
21. byte FirstByte = Wire.read(); // First byte of data
22. byte SecondByte = Wire.read();// Second byte of data
24. word rangeReading = word(FirstByte, SecondByte); // Combine two bytes
26. Serial.println(rangeReading); // Prints the reading
27. }
29. }

The working Arduino model was used as a reference to replicate command in Simulink and then on MATLAB. There were several hindrances while executing the model in Simulink and MATLAB. The Simulink diagnostic viewer did not return any error, it stopped working after few seconds which made it difficult to debug. The second option was to use the MATLAB command window to replicate Arduino’s working model.

1. clc                                        % Clears command window
3. fprintf('Cleared workspace\n');            % Displays message in command window
4. a = arduino('com4','uno','libraries','I2C')% Creates an arduino object 'a'
5. scanI2CBus(a);
6. mb1242 = device(a, 'I2CAddress', '0x70');  % Creates slave object 'mb1242'
8. fprintf('Initialisation Complete \n');     % Initial setup complete message

11. **for** i = i:50                                % Creates a loop to read no: of readings
12. writeRegister(mb1242, 1, hex2dec('51'),  'uint8');  % WRITE request to **register** '0x51'
13. pause(0.1);                             % 100mS delay - Important
14. data = read(mb1242, 2, 'uint8');        % Sends a READ request the SLAVE 0x70 **for** 2 Bytes of data
15. bitconcat(fi(data(1),0,8,0), fi(data(2), 0, 8, 0))% Received 2 Bytes are combined
16. end
18. fprintf('End of program\n');                % Displays message in command window
19. % clear all
20. clear a                                     % Clears Arduino object 'a' and associated data
21. clear data
22. clear i
23. clear mb1242

Debugging and interfacing MB1242 with MATLAB/Simulink

Debugging was carried out mainly by using a logic analyser. The code written in MATLAB was successfully implemented with the help of logic analyser.

