Student Name: **I**ssa **G**abriel **M**alke

E-Mail: issa.malke@ufl.edu

TAs: Andy Gray
Jake Easterling
Ralph F. Leyva

Instructors: Dr. A. Antonio Arroyo

Dr. Eric M. Schwartz

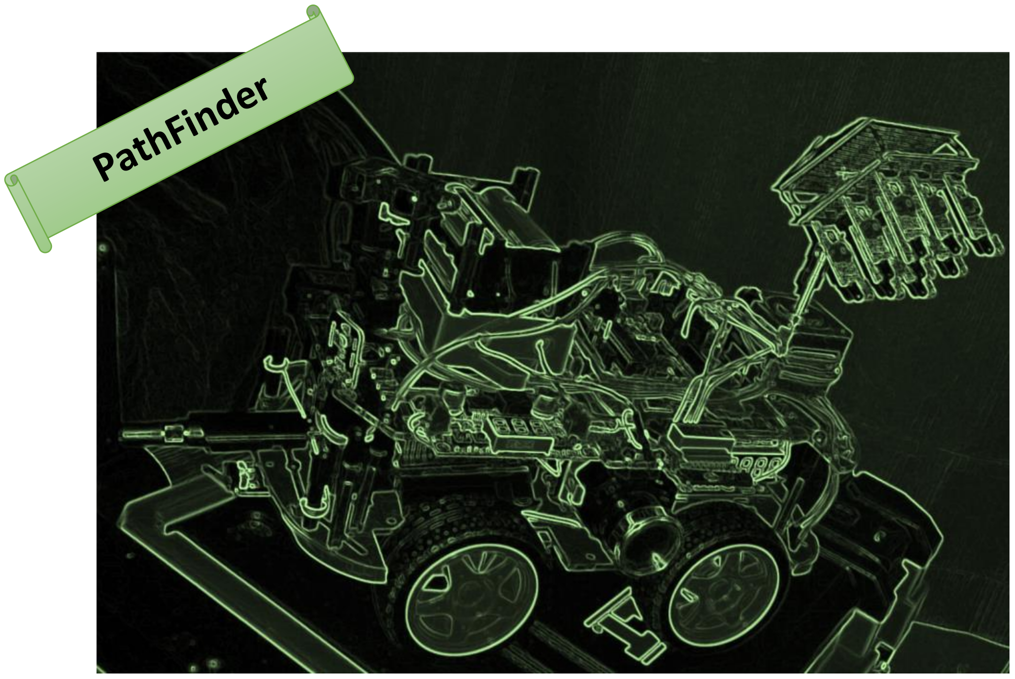
University of Florida

Department of Electrical and Computer Engineering

EEL 4665/5666

Intelligent Machine Design Laboratory

Written Report 1



A Robot that Find its way through obstacles, and map the path to follow it back to where it came from.

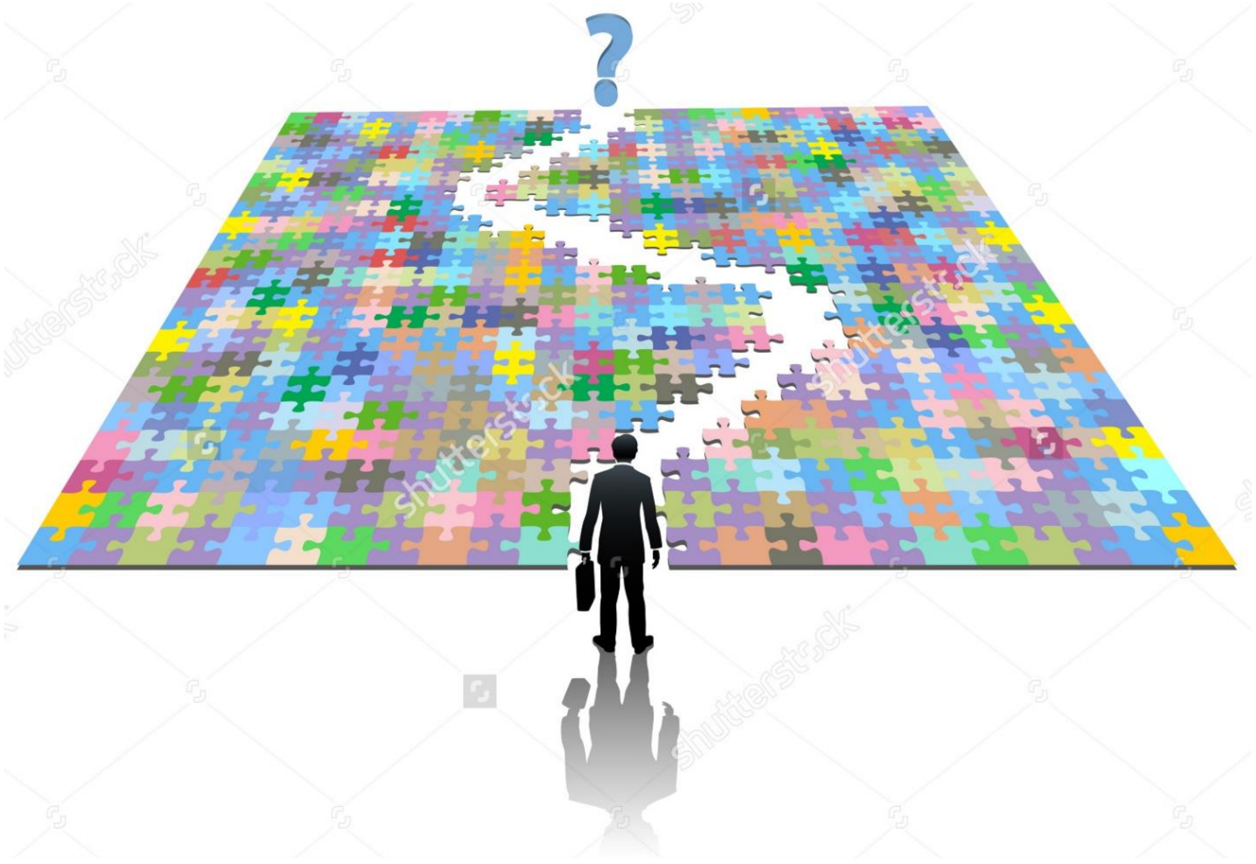
Table of Content

Abstract:

The name of the Robot reflects its duty which is find its way through obstacles, and map the path to follow it back to where it came from. In other words, the Robot should find its way from a starting point to an end point "specified at x distance" then follow the same path to get back to where he came from "starting point". To achieve such a task, the Robot should try to avoid any obstacle blocking his way, and save the path he followed "mapping" so he can take the same path to get back to where he started from.

Introduction:

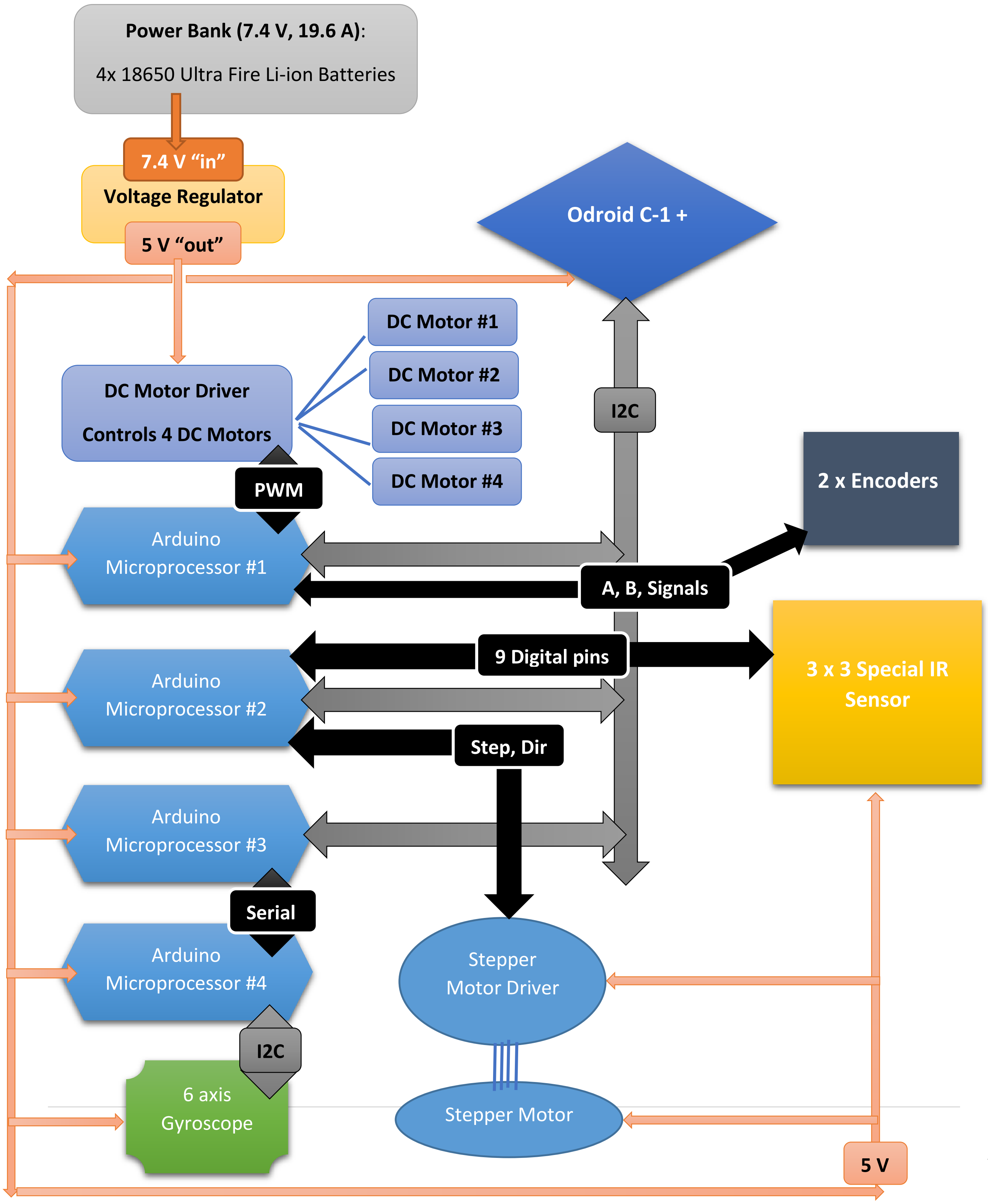
In many Robot Competitions the Robot is required to cross an obstacles field to reach a specific point then turn back to where it came from. Also, in real world, many tasks required to drive to a specific point where we stay for a while to do something then we head back to where we came from. Now a days, an autonomous vehicle is being design that can handle such a task.



So that what influenced me to come up with my idea about designing & implementing a Robot that can achieve that task, which is crossing obstacles field to get to a point (x,y), map the path way, and used it to return to where it came from. So **PathFinder** was born! ☺

Integrated System:

The following is my Robot's design flow charts that describes the overall system.



The explanation of the above Diagram:

- Power Bank: It consist of four 18650 3.7 v 9800 mah Ultra Fire Li-ion Batteries, that each pair is connected in series to get the 7.4 v and then the two pairs are connected in parallel to get the 19.6 A.
- Voltage Regulator: Is used to step down the input voltage to 5 v so it is safe for all circuits' components, and operable.
- DC Motor Driver: H bridge cd motor driver is used to control the four dc motors by thee pwm signals from an Arduino Microprocessor.
- Arduino Microprocessor #1: is used to process the two signals A & B that are coming from two encoders to determine the direction and the speed of the Robot. Also, this microprocessor is used to control the DC Motor driver.
- Arduino Microprocessor #2: is used to control the stepper motor by sending the direction & the step(s) to the stepper motor driver. Also, is used to process thee 9 digital signals thee are coming from the Special 3 by 3 IR sensor.
- Arduino Microprocessor #3: is used to get the gyroscope data that is being passed through serial port from Arduino Microprocessor #4, and send it to the Odroid C-1 by I2C bus.
- Arduino Microprocessor #4: is used to process the data of the 6 axis gyroscope and send it to the Microprocessor #3 to pass it to Odroid c-1 since the I2C channel of this Microprocessor is used to communicate with the gyroscope sensor.
- Stepper Motor Driver: is used to control the stepper motor so it can rotates in specific steps are required.
- Stepper Motor: is used to rotate the 3 by 3 IR sensor block in specific steps so it can scan bigger area searching for obstacles.
- 3 x 3 Special IR sensor: is used to detect obstacles by sending low digital signal every time it counters an object, this being processed by the Arduino Microprocessor #2. Also, the Odroid c-1 will get notified by I2C bus.
- 6 axis Gyroscope: is connected to the Arduino Microprocessor #4, which will pass its data to Arduino M.P.#3 using serial. This sensor is used to detect any change in the x,y,z axis which will help in mapping & navigation.
- Odroid c-1: THIS IS THE BRAIN of the Robot that is connected to Arduino 1 to 3 by I2C bus so it can get their data and analysis it and make decisions then send orders back to Arduinos to control the DC motors to navigate the Robot.

Mobile Platform & Actuation:

As shown in Figure 1, below, I am using a simple manufactured mobile platform, which comes with four DC motors. I did not look in details about the torque power of the DC motor gear box, which is made of plastic, because the overall weight is not heavy and since I am using four motors then no problem had occurred, so far, and the mobile platform is moving smoothly and fast.

Each pair of motors, two of each side, is connected to one PWM pin and same power line ➔

I only control two pairs not each dc motor individually, which I did on purpose since it is enough to control them that way because if I want to turn right I order the right pair of motors to move forward while at the same time the left pair of motor will move backwards.

All the parts of the mobile platform is made of hard plastic, including the wheels and their gear boxes, which is sufficient for this project.

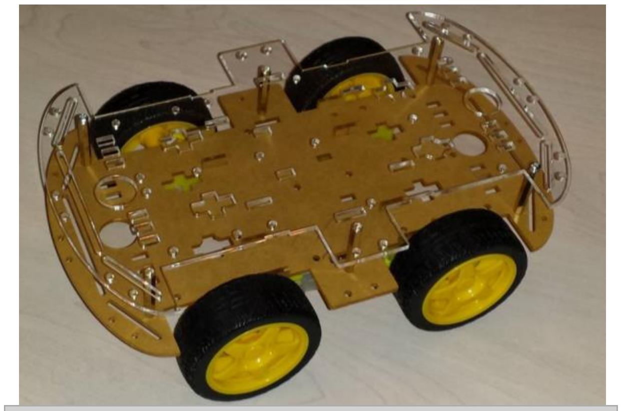


Figure 1, Mobile Platform

Sensors:

A special Sensor 3 by 3 IR Matrix is designed using 9 IR sensors "Figure 2 & 3", which has a range from 2 to 80 cm. Once an obstacle is detected the sensor will inform the microprocessor by sending a low level signal.

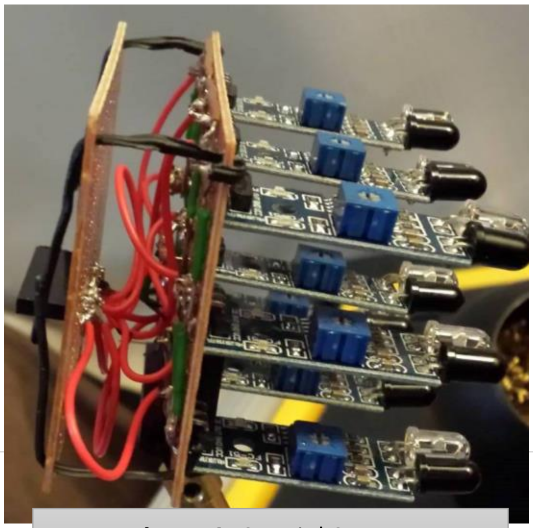


Figure 3, Special Sensor

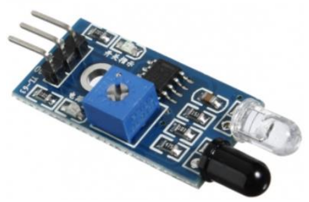


Figure 2, Special Sensor Component

In the Side figure (Figure 4) it shows a 3D module of the mobile platform with the Special Sensor

The range of the IR sensor is reduced to 30-35 cm instead of using its maximum range which is 80 cm to avoid the interference between neighbored IR sensors.

Note that there are three layers of the IR sensors, each layer has 3 IR sensors.

The lowest layer is to detect the ground so it can detect a hole in ground, while the other two layers are used to detect obstacles.

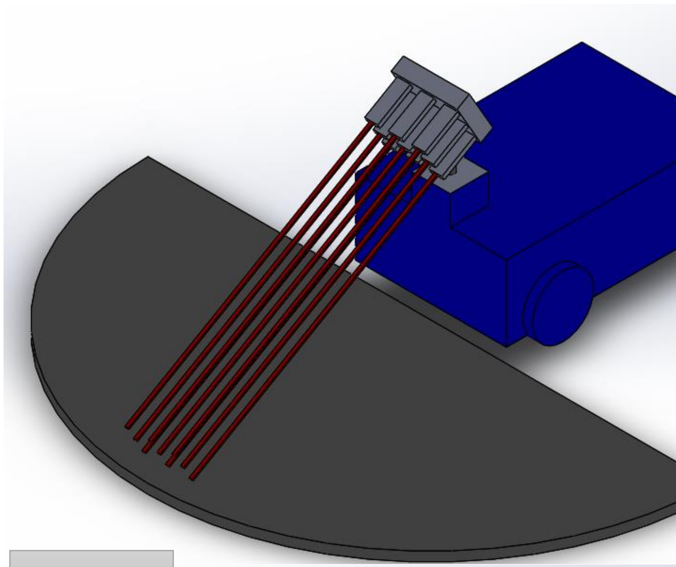
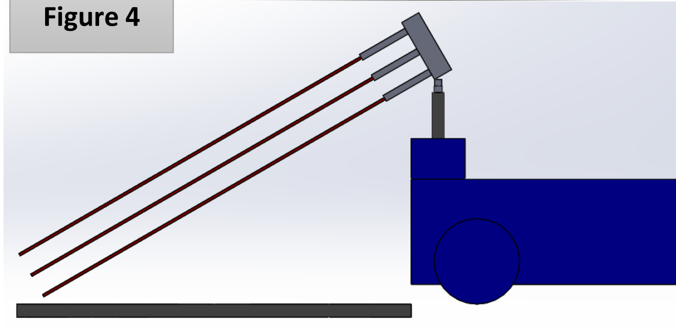


Figure 4



To help the Robot in tracing/detecting its position a 6 axis gyroscope will be used, (Figure 4).

The following figure (Figure 5) shows the 3 axis, and their rotations.

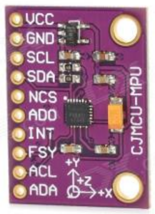


Figure 5, 6 axis Gyroscope

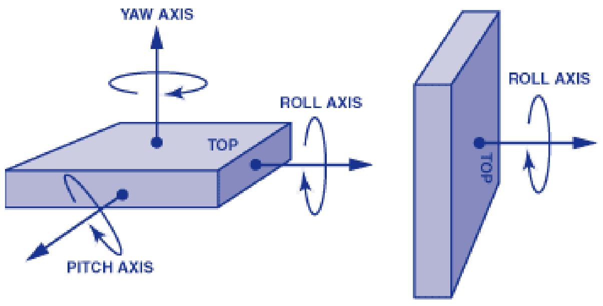


Figure 6

Behaviors:

Let consider the following scenario as in figure (7) ➔ the Robot behaviors, should be as following:

- At the beginning, start scanning the surrounding by rotating the special sensor 180 degrees from one side to another searching for obstacles/ objects.
- Start moving in a straight line until detecting the wall, then turn right "considering the thick wall behind the Robot is facing the Robots' back" keep moving and scanning until there is no wall detected on the Robots' side, which means is safe to turn to that side.
- Once the Robot is turned left it will move forward, and since the dimensions of the arena "should be known to the Robots' database" is fixed "not open space" the Robot will calculate how far has gone so it knows that it is time to move left since the goal point is place on its left.
- After turning left the Robot will continue straight till it detecting the wall and since the wall is connected to the side wall then the Robot will try the other way so it will turn left, the Robot prioritize (1)forward, (2)right or left turns (3)backwards.
- Again facing another wall will force the robot to make a turn and this time to the right. At this point the Robot knows that the goal/ destination point is nearby, and since the Robot know its coordinates (x,y) he will turn towards it and move till he get there.

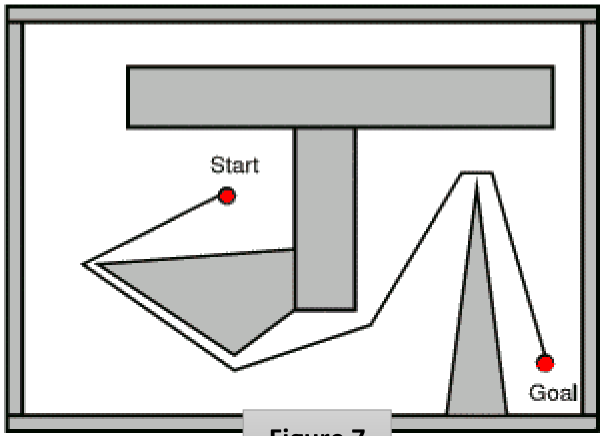


Figure 7

Conclusion:

I am really excited for this project, but at the same time it is a big challenge to map the path way so the Robot can use it to return to its starting point. I will do my best and try to be creative and enjoy the experience!

Appendix A:

Motor Drive Arduino Code:

```
#include <Wire.h>

// Global Variables
volatile char command, Right, Left;

volatile int Gyro_X =0;
volatile int Gyro_Y =0;
volatile int Gyro_Z =0;

unsigned long last_motor_write = 0;
boolean motors = false;

//M1
const int left_pwm = 3;
const int left_dira = 4;
const int left_dirb = 5;

//M2
const int right_pwm = 6;
const int right_dira = 7;
const int right_dirb = 8;

void Move_Forward(){

  digitalWrite(left_dira, HIGH);
  digitalWrite(left_dirb, LOW);
  digitalWrite(right_dira, HIGH);
  digitalWrite(right_dirb, LOW);
}
void Turn_Left(){

  digitalWrite(left_dira, LOW);
  digitalWrite(left_dirb, HIGH);
  digitalWrite(right_dira, HIGH);
  digitalWrite(right_dirb, LOW);
}
void Turn_Right(){

  digitalWrite(left_dira, HIGH);
  digitalWrite(left_dirb, LOW);
  digitalWrite(right_dira, LOW);
  digitalWrite(right_dirb, HIGH);
}
void Move_Backward(){

  digitalWrite(left_dira, LOW);
  digitalWrite(left_dirb, HIGH);
  digitalWrite(right_dira, LOW);
  digitalWrite(right_dirb, HIGH);
}
void Turn_Off(){

  digitalWrite(left_dira, LOW);
  digitalWrite(left_dirb, LOW);
  digitalWrite(right_dira, LOW);
  digitalWrite(right_dirb, LOW);
}

void setup() {

 // Serial.begin(9600);
  Serial.begin(115200);
  //setup i2c
  Wire.begin(4); //slave with address of 4
  Wire.onReceive(Move);
  Wire.onRequest(returnGyro);

  //setup motor pins
  pinMode(left_dira, OUTPUT);
  pinMode(left_dirb, OUTPUT);
  pinMode(right_dira, OUTPUT);
  pinMode(right_dirb, OUTPUT);
}
int8_t Gyro_buffer[3]; // The Buffer, which will store the INcoming Data
void loop() {
  // put your main code here, to run repeatedly:
  unsigned long curr_time = millis();
 /*
  if(curr_time - last_motor_write > 1000){
    analogWrite(left_pwm, 0);
    analogWrite(right_pwm, 0);
    }

  while(Serial.available()){
   /*
    Read INcoming Data Byte by Byte
    U may Use Serial.parseInt() to read Int(s), but on the sender use Serial.println();
    USING A BUFFER (in both sender, and receiver)IS WAY FASTER! & BETTER!, but use high
freq. "115200"
   */
   /*
   int bytes = Serial.available();
   for(int i = 0; i <= bytes; i++){
   Gyro_buffer[i] = Serial.read();
  // parse buffer for your string/command
   }
    // Read INcoming Data As a Buffer
   Gyro_X = Gyro_buffer[0];
   Gyro_Y = Gyro_buffer[1];
   Gyro_Z = Gyro_buffer[2];
  }
  Serial.print(Gyro_X);
  Serial.print(Gyro_Y);
  Serial.println(Gyro_Z);
  delay(15);
  */
}

void Move(int howMany){

  last_motor_write = millis();

  command = Wire.read(); //get first byte - command
  Left = Wire.read();
  Right = Wire.read();

  if(command == 3){
    motors = false;
    //Move Backward
    Move_Backward();
    analogWrite(left_pwm, 0);
    analogWrite(right_pwm, 0);
  }
  if(command == 0){
    motors = true;
    //left forward, right backward
    Move_Forward();
    analogWrite(left_pwm, Left);
    analogWrite(right_pwm, Right);
  }
  if(command == 2){
    motors = true;
    //left backward, right forward
    Turn_Right();
    while(Gyro_Z == 0){
    analogWrite(left_pwm, Left);
    analogWrite(right_pwm, Right);
    }
  }
  if(command == 1){
    motors = true;
    //left backward, right backward
    Turn_Left();
    analogWrite(left_pwm, Left);
    analogWrite(right_pwm, Right);
  }
}
void returnGyro(){

     // uint8_t Gyro_buffer[3] = {Gyro_X, Gyro_Y, Gyro_Z};
        uint8_t Gyro_buffer[3];
            Gyro_buffer[0] = Gyro_X ;
            Gyro_buffer[1] = Gyro_Y;
            Gyro_buffer[2] = Gyro_Z ;

      Wire.write(Gyro_buffer, 3);
    }
```