

Wireless Motion Controlled Robot

Introduction / Description:

Our project is designing and building a motion wireless controller where the motors' speed and position is controlled by the motion of a remote building off of our control of the robot using an arduino in lab 4 where the connections between the arduino and motors remained unchanged. An arduino board (Elegoo EL-CB-001 UNO R3), accelerometer (MPU 6050), and two RF bluetooth transceivers (HC-05) are the additional sensors that were primarily used for this project. The functionality of each component are:

The Elegoo EL-CB-001 UNO R3 arduino will be used for the remote control and powered by a 9V battery pack where we included a switch to control the wireless connection to arduinos.

LED's were also incorporated into the remote control such that they were direction indicators.

Moreover, we used two RF bluetooth sensors that'll be connected to UART ports of the arduino boards. Furthermore, the RF bluetooth (HC-05) acts as a transmitter where it'll transmit signals from the remote to the other RF bluetooth (HC-05) on the motor's arduino (it'll behave as a receiver). The accelerometer (MPU 6050) role in this project was to read movement and provide data to the remote control arduino, which will send commands to the motors that are dependent on the accelerometer values. tilting the accelerometer forwards or downwards controls the speed. We're controlling both of the speed and direction of the wheels via the accelerometer through a bluetooth connection. If we tilt it forwards, it speeds up the wheels' motors, however, by tilting it downwards it decelerates and stop the wheels' motors. In terms of direction, the motors' wheels

CW or CCW rotations are controlled by move the accelerometer left and right. Our block diagram in the following page may give you a visual representation of our project.

Block Diagram / Schematics:

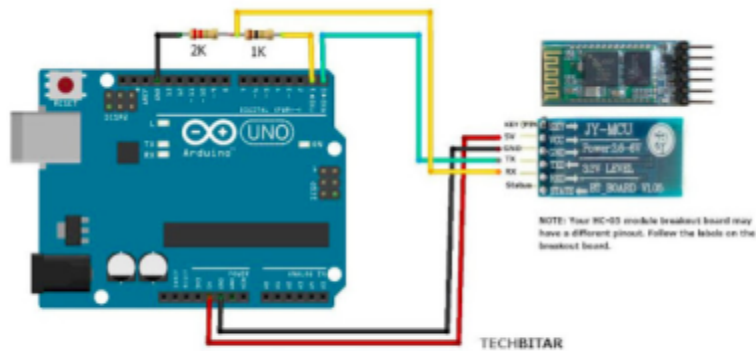


Fig. 1: Bluetooth Schematic Diagram

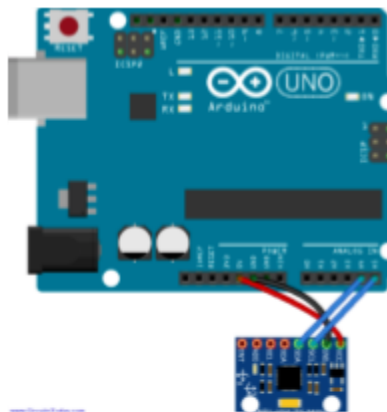


Fig. 2: Accelerometer schematics Diagram

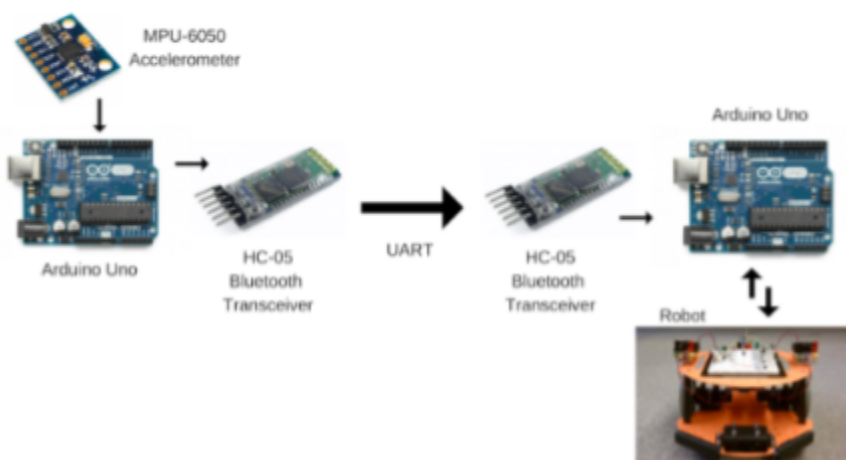


Fig. 3: Block Diagram of the wireless controller design

Design Process:

We have split the final project into two processes:

1. Hardware design
2. Software design
3. Additional sensors and actuators

Hardware Design:

For the hardware design, we followed our block diagram and the necessary schematics for each sensor (e.g. bluetooth and accelerometer) such that we built two sub-circuits for this project:

1. Remote control's sub-circuit
2. Robot's sub-circuit

Controller's wire connections:

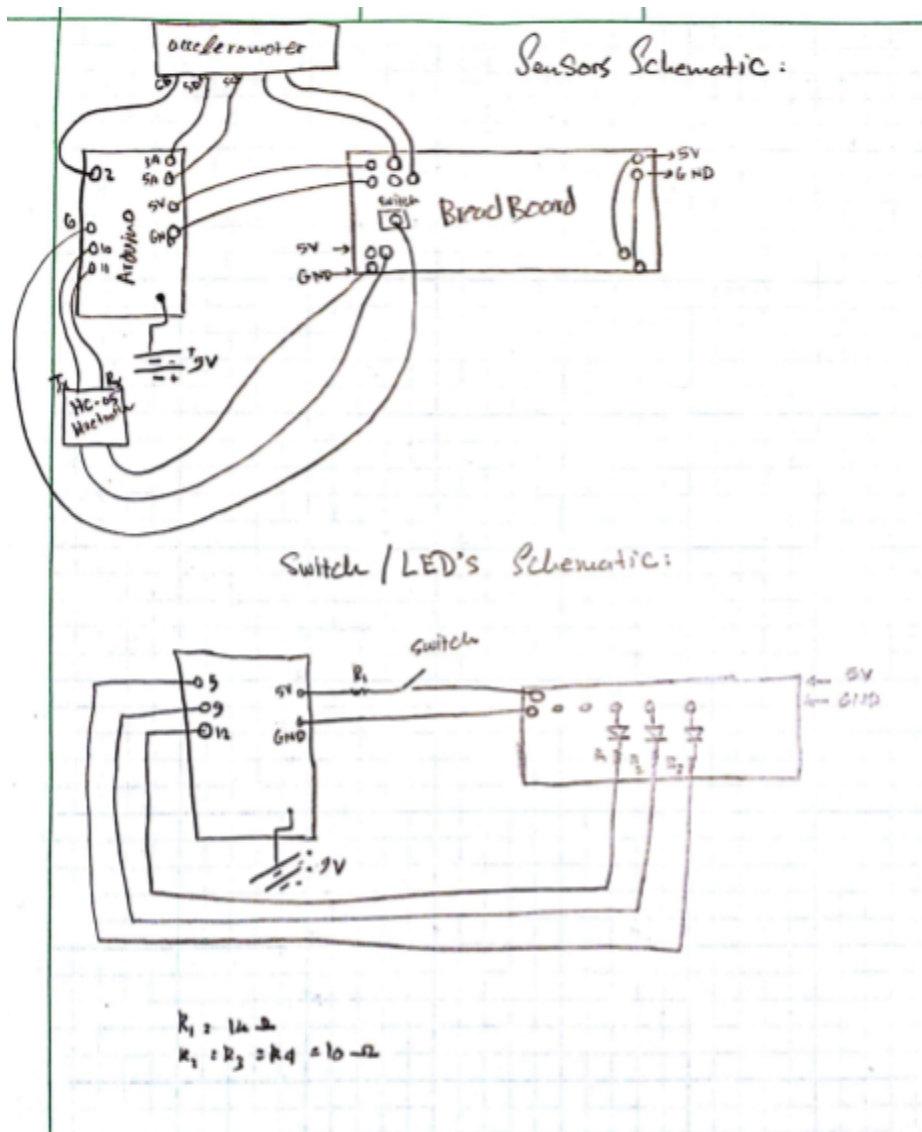


Fig. 4: wire connections of the controller with all the sensor connections

Robot's additional circuit wire connections:

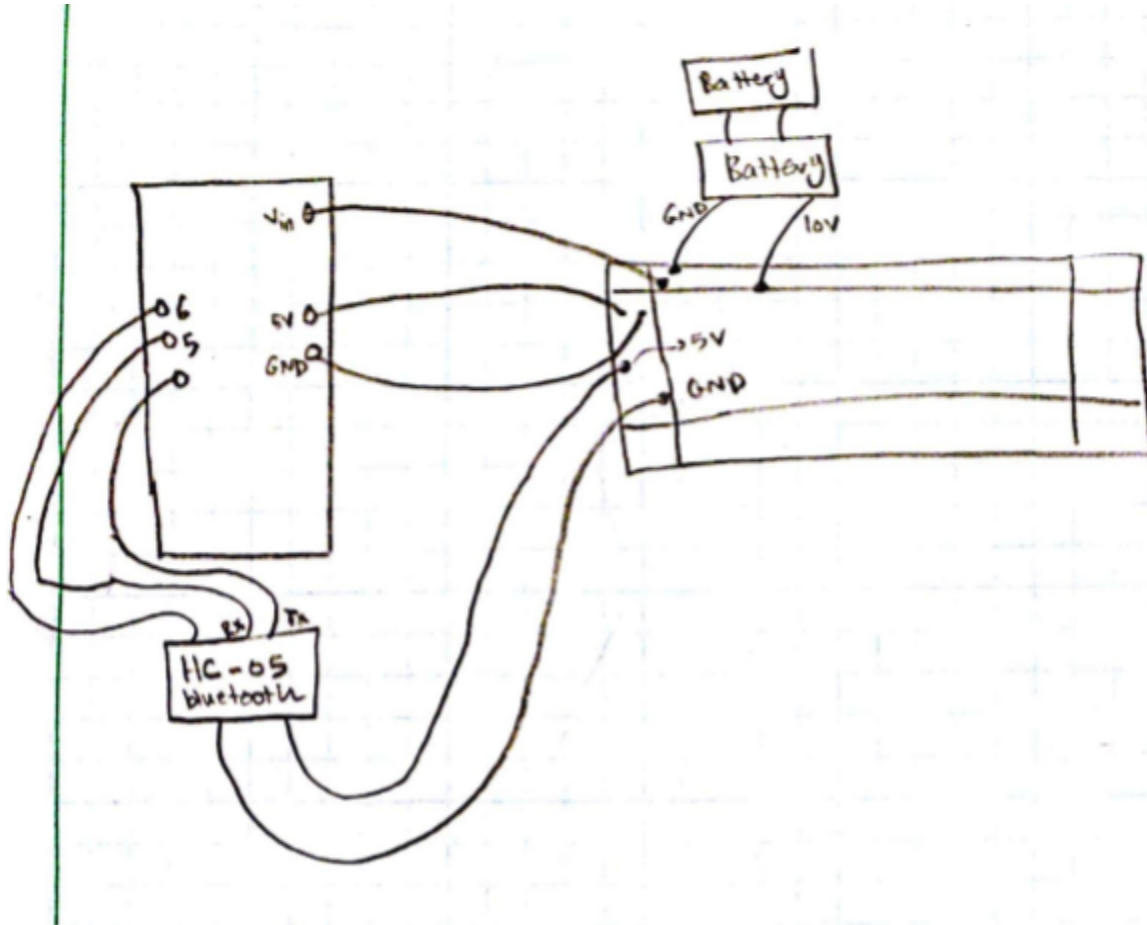


Fig. 5: Robot's power connection and bluetooths connection schematic

Sensors functionalities:

1. Bluetooth (HC-05)

A bluetooth module is a:

- serial port module where its communication is via serial communication
- It can be set to either slave or master mode in comparison to other bluetooth sensors

(Meaning, it can communicate with seperate arduino boards)

The previous diagrams (see Fig. 1 & 5) indicate the hardware wire connections of the HC-05 bluetooth and the functions of each pin which are discussed below in Table. 1:

Pin	Description	Function
VCC	+5V	Connect to +5V
GND	Ground	Connect to Ground
TXD	UART_TXD, Bluetooth serial signal sending PIN	Connect with the MCU's (Microcontroller and etc) RXD PIN.
RXD	UART_RXD, Bluetooth serial signal receiving PIN	Connect with the MCU's (Microcontroller and etc) TXD PIN.
KEY	Mode switch input	If it is input low level or connect to the air, the module is at paired or communication mode. If it's input high level, the module will enter to AT mode.

Table. 1: Pin functionalities of the HC-05 Bluetooth (online source)

After setting up the necessary hardware wire connection of the HC-05 bluetooth (see Fig. 1 & Table. 1) onto the arduino board and completing the source code for it's functionality into the arduino board, we'll need another bluetooth sensor that'll communicate with the Elegoo EL-CB-001 UNO R3 arduino (see Fig. 6).

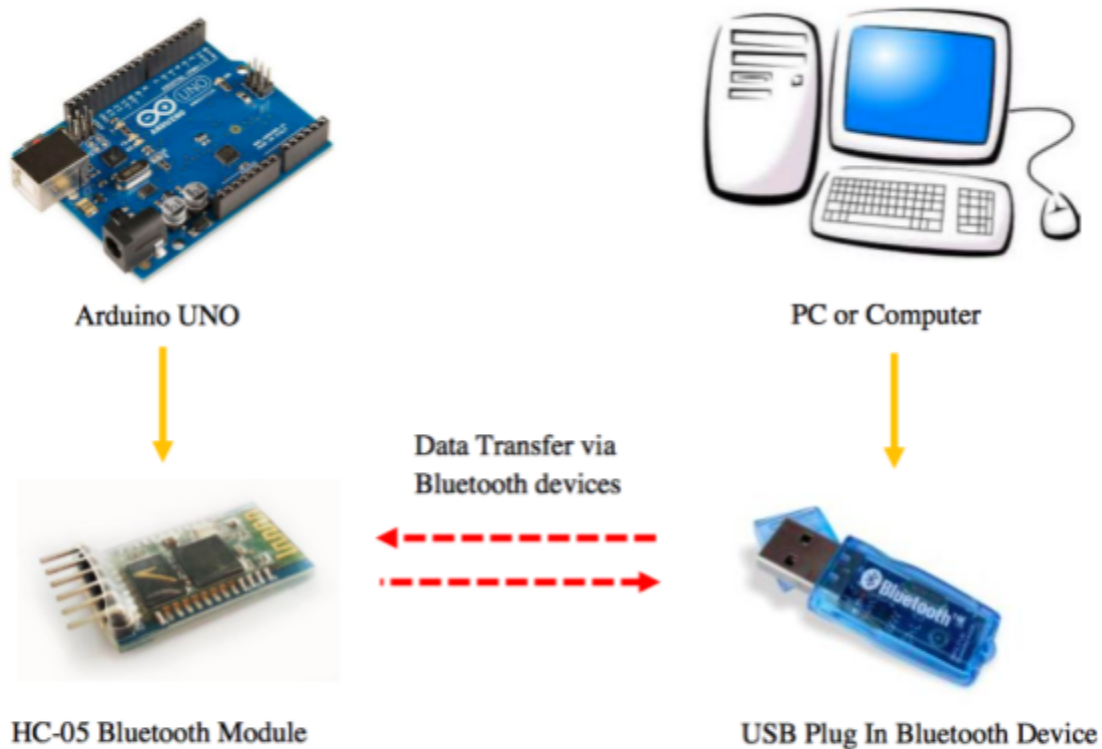


Fig. 6: Block diagram of wireless bluetooth transmission

Therefore, in our case we connected the UNO arduino board, that's controlling the wheels, with the other HC-05 bluetooth sensor in order to achieve data transfer between bluetooth devices. To connect the two devices, they needed to first be put into AT mode and paired with each other through a series of commands. A tutorial for connecting the devices can be found here:

<http://www.martyncurrey.com/connecting-2-arduinos-by-bluetooth-using-a-hc-05-and-a-hc-06-pair-bind-and-link/>

2. Accelerometer

The accelerometer is a:

- IMU sensor
- It's based on MEMS
- It communicates via the I2C protocol

The IMU sensor has a variety of applications and the accelerometer is one of those applications such that it gets the position values (which are usually at an angle) of the object. In short, the sensor detects the positions of different objects. And the MEMS is just the microscopic devices especially for moving parts.

The MPU 6050 accelerometers consists of all the previous components where it works on the principle called piezoelectric effect which depends on the current produced by the walls as we tilt the box and the ball is compressed onto one of the walls (see Fig. 7 for a visual).

Furthermore, the MPU 6050 accelerometer communicates via an I2C protocol Which is a two wire interface (SCL (serial clock) and SDA (serial data)) to connect low speed devices. By implementing the wire connections seen in Fig. 2, the program will use the arduinos interrupt pin. The next stage would be coding and uploading in the required commands and functions into the arduino that would, in our case, communicate well with the wireless bluetooth connections and arduino boards.

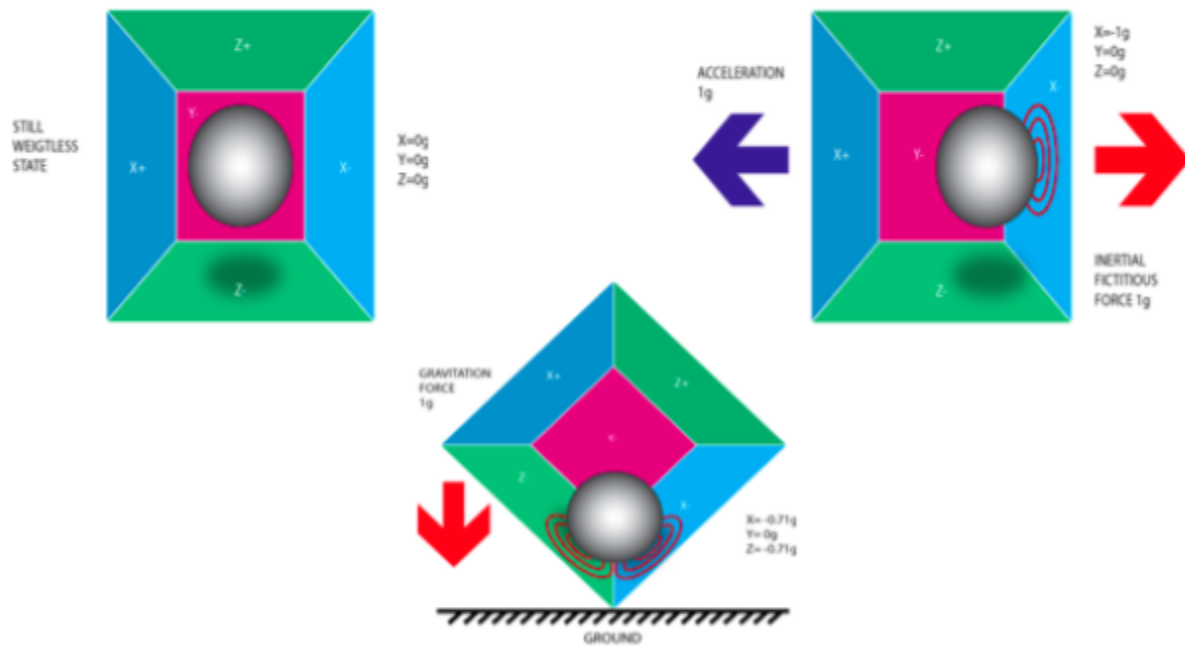


Fig. 7: Piezoelectric effect within the accelerometer

Remote Control CAD Design:

We've also designed a remote on solidworks that consists of the remote control circuitry indicated in Fig. 4 which made the navigational and control process of the robot a little easier (See Fig. 8 & 11).

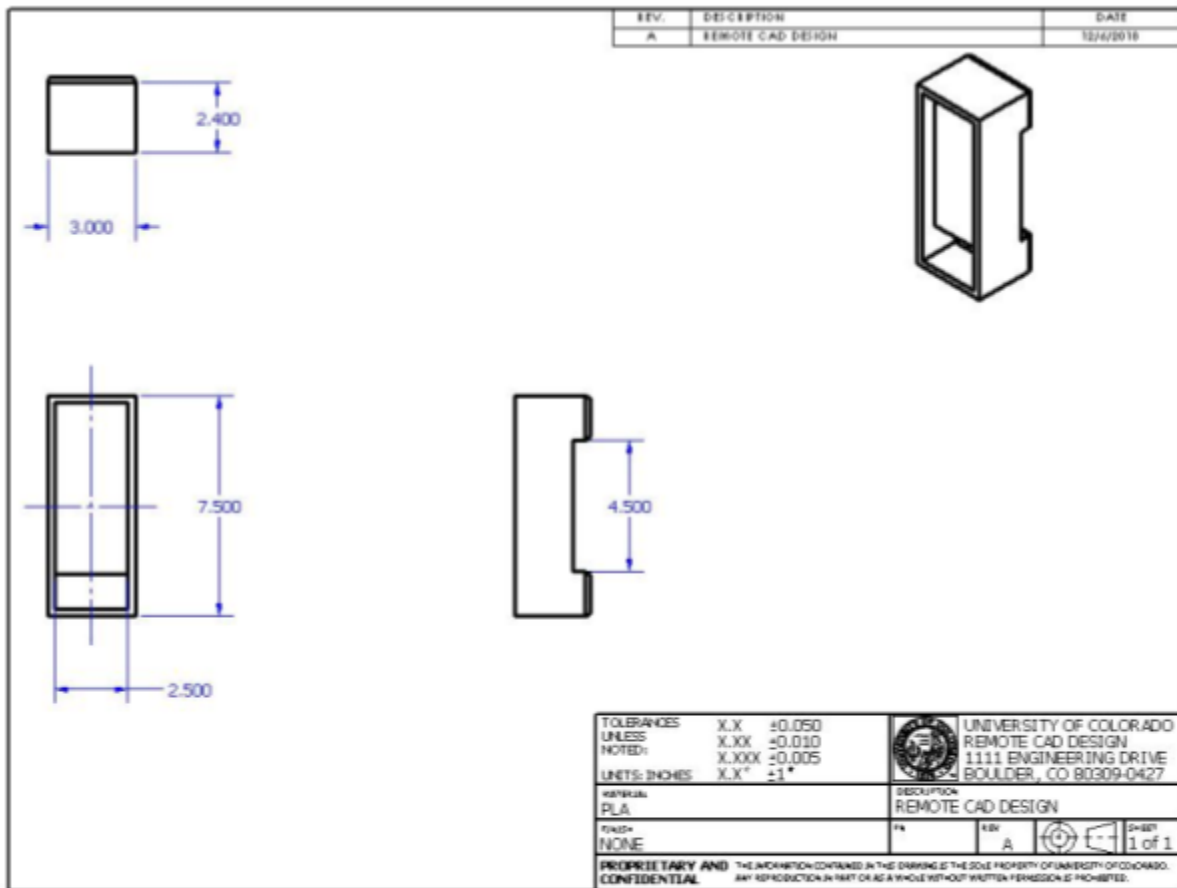


Fig. 8: Engineering Drawing of Remote Control

Software Design:

For the software design, we programmed the necessary functions into two arduinos, where we controlled both of the speed and direction of the wheels. The function of one of the arduinos was to sample accelerometer data and transmit instructions to the second arduino on the robot, which directly operated the motors. If we tilt the accelerometer forwards, it speeds up the wheels' motors and by tilting it backwards it decelerates and stops the wheels' motors. In terms of

direction, the motors' wheels CW or CCW rotations are controlled by tilting the accelerometer left and right. The sensitivity of the accelerometer made it difficult to stop turning left or right so we implemented functions in our software that check when the accelerometer is moved back into a neutral position after turning. This prevents the motion of tilting the controller back into a neutral position from commanding the robot to turn the opposite direction that it was turning in previously. We used two separate sets of code for each arduino. The following is an explanation of the main components of our software (The full code is very long). The entire code, which includes the declaration of variables and the setup for the arduinos can be found in the appendix.

Motion Sensing Remote Code:

Main Loop:

```
void loop()
{
  delay(200);

  Wire.beginTransmission(MPU_addr);

  Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)

  Wire.endTransmission(false);

  Wire.requestFrom(MPU_addr,14,true); // request 14 registers

  AcX=Wire.read()<<8|Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcY=Wire.read()<<8|Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcZ=Wire.read()<<8|Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  Tmp=Wire.read()<<8|Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
  GyX=Wire.read()<<8|Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyY=Wire.read()<<8|Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyZ=Wire.read()<<8|Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)

  //Serial.print("AcX = "); Serial.print(AcX);
```

```

//Serial.print(" | AcY = "); Serial.print(AcY);

//Serial.print(" | AcZ = "); Serial.print(AcZ);

//Serial.print(" | GyX = "); Serial.print(GyX);

//Serial.print(" | GyY = "); Serial.println(GyY);

//Serial.print(" | GyZ = "); Serial.println(GyZ);

//delay(300);

if (digitalRead(pinON) == LOW) { //checks the state of the switch. Low: power off

    digitalWrite(13,LOW);

    BTserial.write(stp); //send stop instruction

    digitalWrite(LEDLeft,LOW); //direction leds are turned off

    digitalWrite(LEDRight,LOW);

    digitalWrite(LEDForward,LOW);

}

do {} while (digitalRead(pinON) == LOW); //wait until switch is high to begin sending instructions

digitalWrite(13,HIGH); //turn on led to indicate motion control is active

if(GyX >= 11000) { //accelerometer is tilted right

    Serial.println("Right");

    BTserial.write(right); //send right turn instruction

    digitalWrite(LEDForward,LOW); //led indicating right is turned on, others are turned off

    digitalWrite(LEDLeft,LOW);

    digitalWrite(LEDRight,HIGH);

    delay(400);

    checkLeft(); //call function that checks when remote is returned to neutral position.

} else if(GyX <= -11000) { //left tilt

    Serial.println("Left");

    BTserial.write(left); //send left turn instruction

    digitalWrite(LEDForward,LOW); //led indicating left is turned on

    digitalWrite(LEDRight,LOW);

    digitalWrite(LEDLeft,HIGH);

```

```

    delay(400);

    checkRight(); //call function that checks when remote is returned to neutral position.
}

else if(GyY >= 5500) { //forward tilt

    //Serial.println("forward");

    Serial.println("Fast");

    BTserial.write(fast); //send forward instruction

    digitalWrite(LEDLeft,LOW); //led indicating forward is turned on

    digitalWrite(LEDRight,LOW);

    digitalWrite(LEDForward,HIGH);

    delay(300);

} else if(GyY <= -5000) { //backward tilt

    Serial.println("Slow");

    BTserial.write(slow); //send slow down instruction

    digitalWrite(LEDLeft,LOW); //all leds turned off

    digitalWrite(LEDRight,LOW);

    digitalWrite(LEDForward,LOW);

    delay(600);

}

}

```

In addition to the main loop, the program includes two other functions: `checkLeft()` and `checkRight()`.

The purpose of these functions is to stop the main loop and wait until the remote has been tilted back into a neutral position. This allows the user holding the remote to accurately control the robot by tilting the remote back to its starting position before picking a new direction to tilt the remote. These two functions are almost identical and look like this:

checkLeft Code:

```
void checkLeft() {  
  
    delay(200);  
  
    while(1) { //endless while loop that terminates through break statements  
  
        Wire.beginTransmission(MPU_addr);  
  
        Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)  
  
        Wire.endTransmission(false);  
  
        Wire.requestFrom(MPU_addr,14,true); // request a total of 14 registers  
  
        AcX=Wire.read()<<8|Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)  
  
        AcY=Wire.read()<<8|Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)  
  
        AcZ=Wire.read()<<8|Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)  
  
        Tmp=Wire.read()<<8|Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)  
  
        GyX=Wire.read()<<8|Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)  
  
        GyY=Wire.read()<<8|Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)  
  
        GyZ=Wire.read()<<8|Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)  
  
        if(GyX <= -11000) { //accelerometer has been tilted back to neutral position  
  
            BTserial.write(slow); //stop turning  
  
            digitalWrite(LEDForward,LOW); //led indications for direction are turned off  
  
            digitalWrite(LEDRight,LOW);  
  
            digitalWrite(LEDLeft,LOW);  
  
            delay(400);  
  
            break; //break out of the loop and return  
  
        }  
  
        if (digitalRead(pinON) == LOW)  
  
            break; //if switch is turned off, stop the robot and return to main loop  
  
    }  
  
    return;  
  
}
```

Motor Driver Code:

Main Loop:

```
void loop()
{

    // Keep reading from HC-06 and send to Arduino Serial Monitor

    if (BTSerial.available()) {

        //Serial.write(BTSerial.read());

        prev_state = state;

        state = BTSerial.read();

    }

    // Keep reading from Arduino Serial Monitor and send to HC-06

    //if (Serial.available())

    //BTSerial.write(Serial.read());


    if(state == '0') { //Stop

        digitalWrite(pinCW_Left,LOW); // stop clockwise

        digitalWrite(pinCC_Left,LOW); // stop counter-clockwise

        digitalWrite(pinCW_Right,LOW); // stop clockwise

        digitalWrite(pinCC_Right,LOW); // stop counter-clockwise

        analogWrite(pinSpeed_Left,0);

        analogWrite(pinSpeed_Right,0);

        enc_count_L = 0;

        enc_count_R = 0;

    } else if(state == '1') { //Slow Down

        //if((L_speed > 5) && (R_speed > 5)) {

        // L_speed = L_speed-5;
```

```

// R_speed = L_speed;

L_speed = 0;

R_speed = 0;


digitalWrite(pinCW_Left,LOW); // stop clockwise
digitalWrite(pinCC_Left,LOW); // stop counter-clockwise
digitalWrite(pinCW_Right,LOW); // stop clockwise
digitalWrite(pinCC_Right,LOW); // stop counter-clockwise

enc_count_L = 0;

enc_count_R = 0;
} else if(state == '2') { //Speed Up
if(prev_state == '3' || prev_state == '4') {

    L_speed = 0;

    R_speed = 0;

    digitalWrite(pinCW_Right,LOW);

    digitalWrite(pinCC_Left,LOW);

}

L_speed = 70;

R_speed = 70;

//Serial.print("forward");

digitalWrite(pinCW_Right,HIGH);

digitalWrite(pinCC_Left,HIGH);

correctWheelDiff(L_speed, R_speed);
} else if(state == '3') { //Turn Right

    digitalWrite(pinCW_Right,LOW);

    digitalWrite(pinCC_Left,HIGH);

    R_speed = 0;

if(prev_state == '4') { //if turning previously, reset speed

    L_speed = R_speed;

```

```

    L_speed = R_speed + 60;
} else {
    L_speed = R_speed + 60;
}
enc_count_L = 0;
enc_count_R = 0;
} else if (state == '4') { //Turn Left
    digitalWrite(pinCW_Right,HIGH);
    digitalWrite(pinCC_Left,LOW);
    //L_speed = 0;
    if(prev_state == '3') { //if turning previously, reset speed
        R_speed = L_speed;
        R_speed = L_speed + 45;
    } else {
        R_speed = L_speed + 45;
    }
    enc_count_L = 0;
    enc_count_R = 0;
}

analogWrite(pinSpeed_Left,L_speed);
analogWrite(pinSpeed_Right,R_speed);
//Serial.write(state);
//delay(500);
}

```

If the two motors of the robot operated in the same way, this main loop would be enough;

however, one motor is more powerful than the other so we included more functions that correct

for this difference by reading the encoder pulses of each motor and adjusting the speed to keep the number of rotations of the motor in sync. The functions for this are the following:

Wheel Speed Correction Code:

```
//Encoder ISRs
```

```
void ISR_count_L(){
```

```
    enc_count_L++;
```

```
    //updateEncoderDiff();
```

```
}
```

```
void ISR_count_R(){
```

```
    enc_count_R++;
```

```
    //updateEncoderDiff();
```

```
}
```

```
void printEncoderVals() {
```

```
    Serial.println("");
```

```
    Serial.print("Left: ");
```

```
    Serial.print(enc_count_L);
```

```
    Serial.print("\t");
```

```
    Serial.print("Right: ");
```

```
    Serial.print(enc_count_R);
```

```
    Serial.print("\t");
```

```
    Serial.print("DIFF: ");
```

```
    Serial.print(enc_diff);
```

```
}
```

```
void correctWheelDiff(int spd_L, int spd_R) {
```

```
    //updateEncoderDiff();
```

```

printEncoderVals();

if(enc_count_L > enc_count_R) { //left wheel needs to be slowed

    analogWrite(pinSpeed_Left,50);

    delay(100);

}

analogWrite(pinSpeed_Left,spd_L); //restore speed

if(enc_count_R > enc_count_L) { //right wheel needs to be slowed

    analogWrite(pinSpeed_Right,50);

    delay(100);

}

analogWrite(pinSpeed_Right,spd_R); //restore speed

return;

}

```

Additional Sensors and Actuators:

Our project was mainly focused on the wireless motion controller, however, we did have some time to add on a few additional components like LEDs and switches. We add a switch to our controller where it connects/disconnects motion control to the robot. Moreover, we incorporated 3 LEDs onto our controller to function as direction indicators. The red LED indicated the left movement, the green LED indicated the forward direction, and the yellow LED indicated the right movement. Fig. 9 indicates the connection diagram of the LEDs and the switch used. And Fig. 4 depicts the incorporation of the LEDs and switch into our controller schematic.

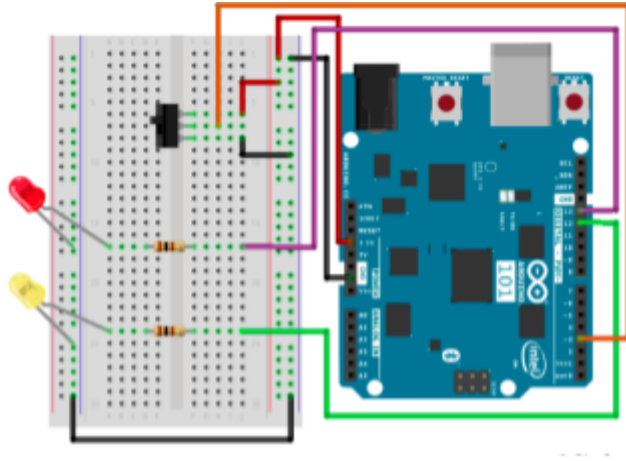


Fig. 9: LED's and switch connection diagram

LED Code:

The following is a sample of how the arduino tells the LEDs to turn on and off:

When the remote is tilted left, the arduino sets the red LED to high and turns the others off.

LEDForward, LEDRight, and LEDLeft have been declared in the setup section of the code and correspond to the pins that the LEDs are connected to.

```
digitalWrite(LEDForward,LOW); //led indicating left is turned on
```

```
digitalWrite(LEDRight,LOW);
```

```
digitalWrite(LEDLeft,HIGH);
```

Switch Code:

The purpose of the switch is to either pause the motion sensing of the remote and stop the motors or allow the remote to transmit instructions to the robot according to how its moved. The following chunk of code is found near the beginning of the main loop:

```
if (digitalRead(pinON) == LOW) { //checks the state of the switch. Low: power off

    digitalWrite(13,LOW);

    BTserial.write(stp); //send stop instruction

    digitalWrite(LEDLeft,LOW); //direction leds are turned off

    digitalWrite(LEDRight,LOW);

    digitalWrite(LEDForward,LOW);

}

do {} while (digitalRead(pinON) == LOW); //wait until switch is high to begin sending instructions

digitalWrite(13,HIGH); //turn on led to indicate motion control is active
```

Experimentation:

To test the functionality of our components, the main tool we used was the arduino serial monitor. This was used to print out accelerometer values to see how movement was represented as data and it was used to view the functionality of the bluetooth modules. The biggest challenge was the bluetooth communication between the two arduinos and the focus of our experiments was to see exactly how data was being transmitted and received. In order to visualize how the communication worked, many print statements were included in the code to display exactly what was being sent by one arduino and what was being received by the other. These can all be found

in the code, commented out as they are not needed after the components have been understood.

Figure 8 shows what this looks like:

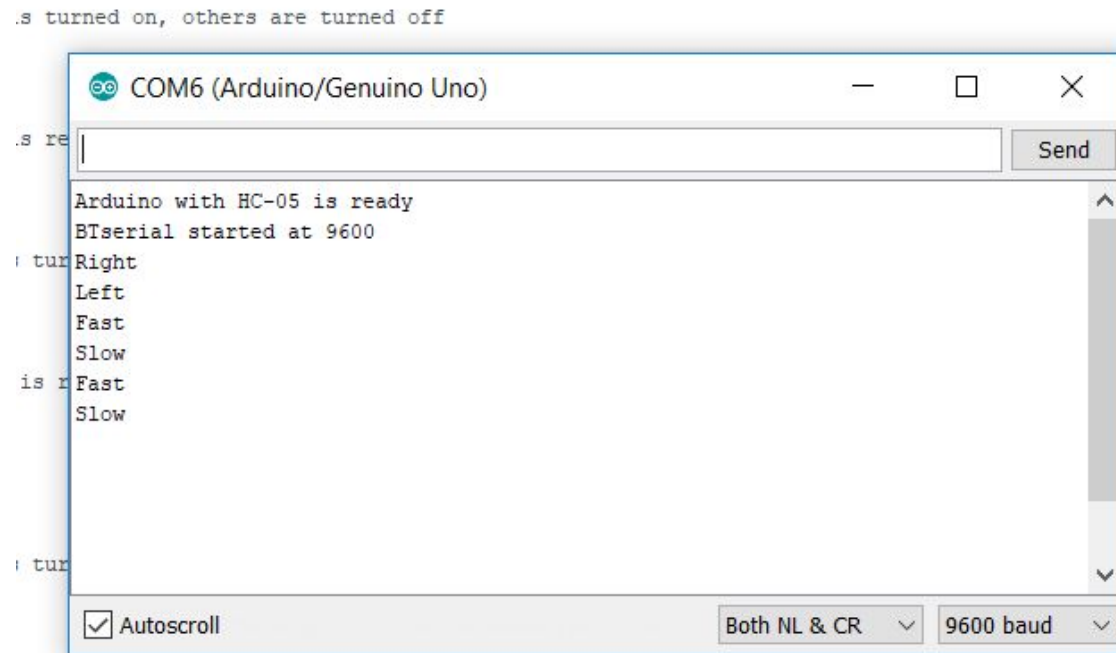


Figure 10: The serial monitor displaying that the bluetooth module is ready, what frequency it is operating at, and the instructions that are being sent when the remote control is tilted in different directions.

In addition to using the serial monitor, the addition of LEDs on the remote let us know what directions the accelerometer was reading. We used three LEDs that each lit up according to how the remote was tilted. This let us know that we were reading the correct values from the accelerometer in our code.

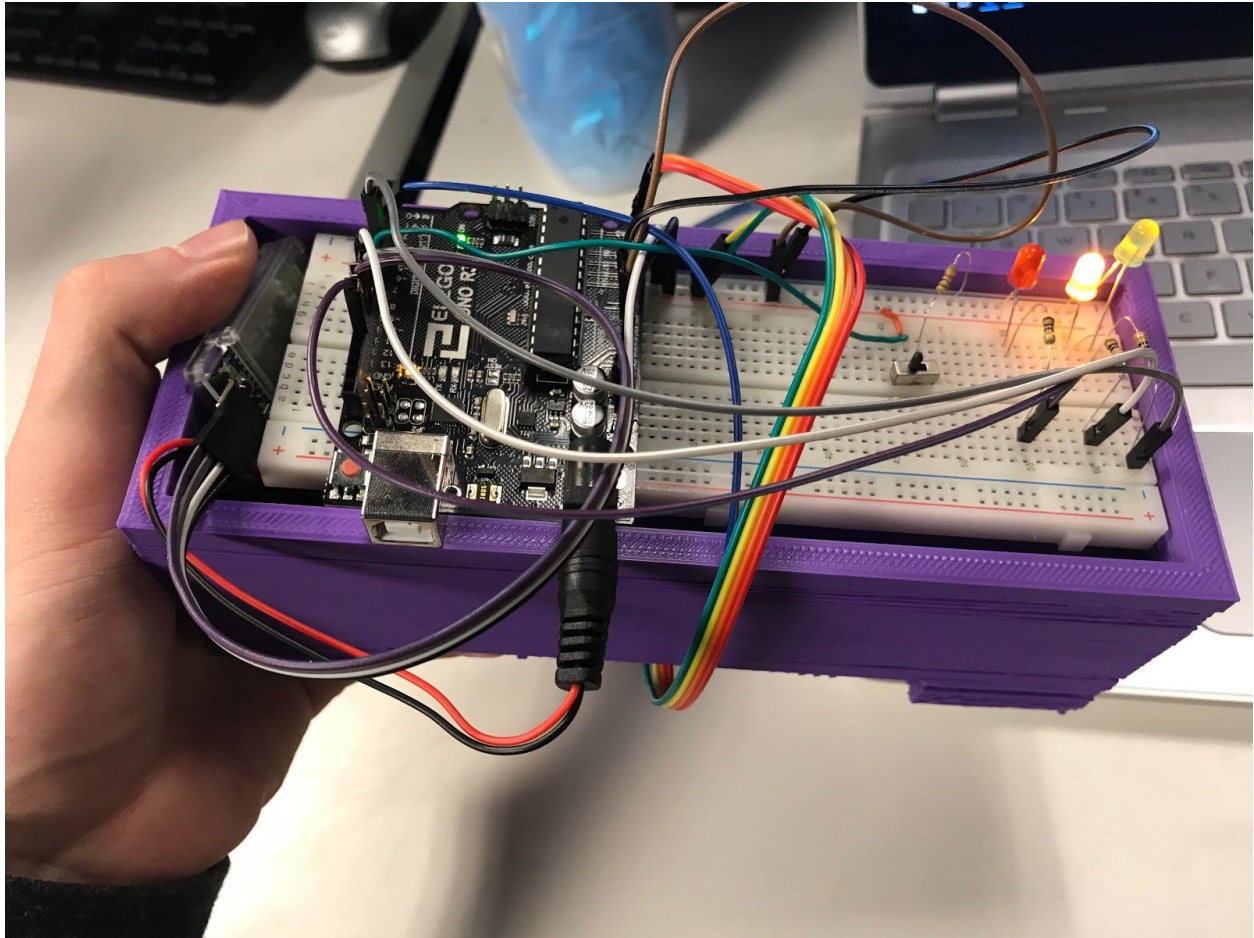


Figure 11: The complete remote with the yellow LED turned on due to a right tilt.

Conclusion:

Project Challenges / Improvements / Future Extensions:

Challenges faced in this project are the following:

1. A wireless controller that's varying in speed such that it's dependent on the degree of our hand tilt.

Unfortunately, our attempt to control the robots' speed via degree of hand tilt was a challenge, therefore, we implemented an improved plan to our project where our wireless controller is

non-varying in speed (so it's independent of the degree of our hand tilt). However, our code does control the robot's speed to some degree where it moves forward when we tilt the controller forwards and stops the motors when we tilt it backwards (see appendix).

2. A wireless controller that's attached to a glove.

As we're testing the functionalities of each sensor, we came to a conclusion that it may be hard for us to properly control the robot if the sensors were connected to a glove since the accelerometer is quite sensitive and inconsistent. The accelerometer is really sensitive. Thus, we designed a remote on SolidWorks that includes some of our components (arduino board, RF bluetooth, and accelerometer) and 3D printed it in order for us to control the robot.

3. Developing a more consistent and reactive code with minimal bugs especially with the right and left movements of the robot.

To improve the ability to control the robot via the accelerometer, we found that slowing down the code and sometimes pausing it until certain conditions were met made our remote the most accurate and easy to use. We used many delay instructions to slow down the arduino and paused the main loop when tilting left or right so that the remote could be returned to the neutral position without reading extra movement.

Improvements and future extensions to our project are the following:

1. A wireless controller that's varying in speed such that it's dependent on the degree of our hand tilt. It could be a coding improvement through the development of more sophisticated algorithms for reading movement or a hardware improvement where we'd use a more accurate accelerometer.

2. The addition of more control elements such as buttons to add more functionality to the robot besides basic movement.

Appendix:

Sources:

Google images:

1. [accelerometer arduino schematic diagram](#)
2. [bluetooth arduinoschematic diagram](#)
3. [switch diagram arduino](#)

Sensor functionalities online manuals:

4. <https://www.gme.cz/data/attachments/dsh.772-148.1.pdf>
5. https://en.wikipedia.org/wiki/Microelectromechanical_systems
6. <https://maker.pro/arduino/tutorial/how-to-interface-arduino-and-the-mpu-6050-sensor>
7. <http://www.martyncurrey.com/connecting-2-arduinosaurs-by-bluetooth-using-a-hc-05-and-a-hc-06-pair-bind-and-link/>
8. <http://i2c.info>

Entire Code:

Motion Control Remote:

```
#include <SoftwareSerial.h>
```

```
SoftwareSerial BTserial(10, 11); // RX | TX
```

```
char c = '';
```

```

char left = '4';

char right = '3';

char fast = '2';

char slow = '1';

char stp = '0';


const int pinON = 6;    // connect pin 6 to ON/OFF switch, active HIGH

const int LEDRight = 9;

const int LEDForward = 5;

const int LEDLeft = 12;


#include<Wire.h>

const int MPU_addr=0x68; // I2C address of the MPU-6050

int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;


void setup()
{
    const int pinON = 6;    // connect pin 6 to ON/OFF switch, active HIGH

    //mpu-6050 setup

    Wire.begin();

    Wire.beginTransmission(MPU_addr);

    Wire.write(0x6B); // PWR_MGMT_1 register

    Wire.write(0);    // set to zero (wakes up the MPU-6050)

    Wire.endTransmission(true);


    // start the serial communication with the host computer

    Serial.begin(9600);

    Serial.println("Arduino with HC-05 is ready");

```

```

// start communication with the HC-05 using 9600

BTserial.begin(9600);

Serial.println("BTserial started at 9600");

pinMode(13,OUTPUT);      // on-board LED

pinMode(LEDForward,OUTPUT);

pinMode(LEDRight,OUTPUT);

pinMode(LEDLeft,OUTPUT);

}

void checkLeft() {

  delay(200);

  while(1) { //endless while loop that terminates through break statements

    Wire.beginTransmission(MPU_addr);

    Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)

    Wire.endTransmission(false);

    Wire.requestFrom(MPU_addr,14,true); // request a total of 14 registers

    AcX=Wire.read()<<8|Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)

    AcY=Wire.read()<<8|Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)

    AcZ=Wire.read()<<8|Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)

    Tmp=Wire.read()<<8|Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)

    GyX=Wire.read()<<8|Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)

    GyY=Wire.read()<<8|Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)

    GyZ=Wire.read()<<8|Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)

    if(GyX <= -11000) { //accelerometer has been tilted back to neutral position

      BTserial.write(slow); //stop turning

      digitalWrite(LEDForward,LOW); //led indications for direction are turned off

      digitalWrite(LEDRight,LOW);

      digitalWrite(LEDLeft,LOW);

      delay(400);

```

```

    break; //break out of the loop and return
}

if (digitalRead(pinON) == LOW)

    break; //if switch is turned off, stop the robot and return to main loop
}

return;
}

void checkRight() {

    delay(200);

    while(1) {

        Wire.beginTransmission(MPU_addr);

        Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)

        Wire.endTransmission(false);

        Wire.requestFrom(MPU_addr,14,true); // request a total of 14 registers

        AcX=Wire.read()<<8|Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)

        AcY=Wire.read()<<8|Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)

        AcZ=Wire.read()<<8|Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)

        Tmp=Wire.read()<<8|Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)

        GyX=Wire.read()<<8|Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)

        GyY=Wire.read()<<8|Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)

        GyZ=Wire.read()<<8|Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)

        if(GyX >= 11000) {

            BTserial.write(slow);

            digitalWrite(LEDForward,LOW);

            digitalWrite(LEDRight,LOW);

            digitalWrite(LEDLeft,LOW);

            delay(400);

            break;

```

```

    }

    if (digitalRead(pinON) == LOW)

        break;

    }

    return;

}

void loop()

{

    delay(200);

    Wire.beginTransmission(MPU_addr);

    Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)

    Wire.endTransmission(false);

    Wire.requestFrom(MPU_addr,14,true); // request a total of 14 registers

    AcX=Wire.read()<<8|Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)

    AcY=Wire.read()<<8|Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)

    AcZ=Wire.read()<<8|Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)

    Tmp=Wire.read()<<8|Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)

    GyX=Wire.read()<<8|Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)

    GyY=Wire.read()<<8|Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)

    GyZ=Wire.read()<<8|Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)

    //Serial.print("AcX = "); Serial.print(AcX);

    //Serial.print(" | AcY = "); Serial.print(AcY);

    //Serial.print(" | AcZ = "); Serial.print(AcZ);

    //Serial.print(" | GyX = "); Serial.print(GyX);

    //Serial.print(" | GyY = "); Serial.println(GyY);

    //Serial.print(" | GyZ = "); Serial.println(GyZ);

    //delay(300);

    if (digitalRead(pinON) == LOW) { //checks the state of the switch. Low: power off

```

```

digitalWrite(13,LOW);

BTserial.write(stp); //send stop instruction

digitalWrite(LEDLeft,LOW); //direction leds are turned off

digitalWrite(LEDRight,LOW);

digitalWrite(LEDForward,LOW);

}

do {} while (digitalRead(pinON) == LOW); //wait until switch is high to begin sending instructions

digitalWrite(13,HIGH); //turn on led to indicate motion control is active

if(GyX >= 11000) { //accelerometer is tilted right

  Serial.println("Right");

  BTserial.write(right); //send right turn instruction

  digitalWrite(LEDForward,LOW); //led indicating right is turned on, others are turned off

  digitalWrite(LEDLeft,LOW);

  digitalWrite(LEDRight,HIGH);

  delay(400);

  checkLeft(); //call function that checks when remote is returned to neutral position.

} else if(GyX <= -11000) { //left tilt

  Serial.println("Left");

  BTserial.write(left); //send left turn instruction

  digitalWrite(LEDForward,LOW); //led indicating left is turned on

  digitalWrite(LEDRight,LOW);

  digitalWrite(LEDLeft,HIGH);

  delay(400);

  checkRight(); //call function that checks when remote is returned to neutral position.

}

else if(GyY >= 5500) { //forward tilt

  //Serial.println("forward");

  Serial.println("Fast");

  BTserial.write(fast); //send forward instruction

```

```

digitalWrite(LEDLeft,LOW); //led indicating forward is turned on

digitalWrite(LEDRight,LOW);

digitalWrite(LEDForward,HIGH);

delay(300);

} else if(GyY <= -5000) { //backward tilt

Serial.println("Slow");

BTserial.write(slow); //send slow down instruction

digitalWrite(LEDLeft,LOW); //all leds turned off

digitalWrite(LEDRight,LOW);

digitalWrite(LEDForward,LOW);

delay(600);

}

}

```

Motor Control Code:

```

#include <SoftwareSerial.h>

SoftwareSerial BTSerial(5, 6); // RX | TX


const int pinCW_Left = 12; //clockwise left

const int pinCC_Left = 11; //counter clockwise left

const int pinSpeed_Left = 10; //speed ref left


const int pinSpeed_Right = 9; //speed ref right

const int pinCC_Right = 7; // counter clockwise right

const int pinCW_Right = 8; // clockwise right


const int encoder_L = 2; //left encoder

const int encoder_R = 3; //right encoder


volatile int enc_count_L = 0;

```

```
volatile int enc_count_R = 0;
```

```
volatile int enc_diff = 0;
```

```
char prev_state = 0;
```

```
char state = 0;
```

```
volatile int L_speed = 0;
```

```
volatile int R_speed = 0;
```

```
void setup()
```

```
{
```

```
    //declare outputs
```

```
    pinMode(pinCW_Left,OUTPUT);
```

```
    pinMode(pinCC_Left,OUTPUT);
```

```
    pinMode(pinSpeed_Left,OUTPUT);
```

```
    pinMode(pinCW_Right,OUTPUT);
```

```
    pinMode(pinCC_Right,OUTPUT);
```

```
    pinMode(pinSpeed_Right,OUTPUT);
```

```
    //declare inputs
```

```
    pinMode(encoder_L,INPUT);
```

```
    pinMode(encoder_R,INPUT);
```

```
    pinMode(13,OUTPUT);      // on-board LED
```

```
    digitalWrite(pinCW_Left,LOW); // stop clockwise
```

```
    digitalWrite(pinCC_Left,LOW); // stop counter-clockwise
```

```
    digitalWrite(pinCW_Right,LOW); // stop clockwise
```

```
    digitalWrite(pinCC_Right,LOW); // stop counter-clockwise
```

```

//analogWrite(pinSpeed_Left,100); // set speed reference, duty-cycle = 100/255

//analogWrite(pinSpeed_Right,100);


//Interrupt init

attachInterrupt(0, ISR_count_L, RISING);

attachInterrupt(1, ISR_count_R, RISING);


Serial.begin(9600);

Serial.println("Arduino with HC-06 is ready");


// HC-06 default baud rate is 9600

BTSerial.begin(9600);

Serial.println("BTserial started at 9600");


//digitalWrite(pinCW_Right,HIGH);

//digitalWrite(pinCC_Left,HIGH);

analogWrite(pinSpeed_Left,0);

analogWrite(pinSpeed_Right,0);

}


//Encoder ISRs

void ISR_count_L(){

    enc_count_L++;

    //updateEncoderDiff();

}


void ISR_count_R(){

    enc_count_R++;

    //updateEncoderDiff();

```

```
}
```

```
void printEncoderVals() {  
    Serial.println("");  
    Serial.print("Left: ");  
    Serial.print(enc_count_L);  
    Serial.print("\t");  
    Serial.print("Right: ");  
    Serial.print(enc_count_R);  
    Serial.print("\t");  
    Serial.print("DIFF: ");  
    Serial.print(enc_diff);  
}
```

```
void correctWheelDiff(int spd_L, int spd_R) {  
    //updateEncoderDiff();  
    printEncoderVals();  
    if(enc_count_L > enc_count_R) { //left wheel needs to be slowed  
        analogWrite(pinSpeed_Left,50);  
        delay(100);  
    }  
    analogWrite(pinSpeed_Left,spd_L); //restore speed  
    if(enc_count_R > enc_count_L) { //right wheel needs to be slowed  
        analogWrite(pinSpeed_Right,50);  
        delay(100);  
    }  
    analogWrite(pinSpeed_Right,spd_R); //restore speed  
    return;  
}
```

```

void loop()

{

    // Keep reading from HC-06 and send to Arduino Serial Monitor
    if (BTSerial.available()) {

        //Serial.write(BTSerial.read());

        prev_state = state;

        state = BTSerial.read();

    }

    // Keep reading from Arduino Serial Monitor and send to HC-06
    //if (Serial.available())

    //BTSerial.write(Serial.read());


    if(state == '0') { //Stop

        digitalWrite(pinCW_Left,LOW); // stop clockwise

        digitalWrite(pinCC_Left,LOW); // stop counter-clockwise

        digitalWrite(pinCW_Right,LOW); // stop clockwise

        digitalWrite(pinCC_Right,LOW); // stop counter-clockwise

        analogWrite(pinSpeed_Left,0);

        analogWrite(pinSpeed_Right,0);

        enc_count_L = 0;

        enc_count_R = 0;

    } else if(state == '1') { //Slow Down

        //if((L_speed > 5) && (R_speed > 5)) {

        // L_speed = L_speed-5;

        // R_speed = L_speed;

        L_speed = 0;

        R_speed = 0;

```

```

digitalWrite(pinCW_Left,LOW); // stop clockwise
digitalWrite(pinCC_Left,LOW); // stop counter-clockwise
digitalWrite(pinCW_Right,LOW); // stop clockwise
digitalWrite(pinCC_Right,LOW); // stop counter-clockwise

enc_count_L = 0;

enc_count_R = 0;

} else if(state == '2') { //Speed Up

if(prev_state == '3' || prev_state == '4') {

    L_speed = 0;

    R_speed = 0;

    digitalWrite(pinCW_Right,LOW);

    digitalWrite(pinCC_Left,LOW);

}

L_speed = 70;

R_speed = 70;

//Serial.print("forward");

digitalWrite(pinCW_Right,HIGH);

digitalWrite(pinCC_Left,HIGH);

correctWheelDiff(L_speed, R_speed);

} else if(state == '3') { //Turn Right

    digitalWrite(pinCW_Right,LOW);

    digitalWrite(pinCC_Left,HIGH);

    R_speed = 0;

if(prev_state == '4') { //if turning previously, reset speed

    L_speed = R_speed;

    L_speed = R_speed + 60;

} else {

    L_speed = R_speed + 60;

```

```

    }

    enc_count_L = 0;

    enc_count_R = 0;

} else if (state == '4') { //Turn Left

    digitalWrite(pinCW_Right,HIGH);

    digitalWrite(pinCC_Left,LOW);

    //L_speed = 0;

    if(prev_state == '3') { //if turning previously, reset speed

        R_speed = L_speed;

        R_speed = L_speed + 45;

    } else {

        R_speed = L_speed + 45;

    }

    enc_count_L = 0;

    enc_count_R = 0;

}

    analogWrite(pinSpeed_Left,L_speed);

    analogWrite(pinSpeed_Right,R_speed);

    //Serial.write(state);

    //delay(500);

}

```