| | | |
|---|---|---|
| Kingdom of Saudi Arabia | | المملكة العربية السعودية |
| Royal Commission at Yanbu | | الهيئة الملكية ينبع |
| University College – Yanbu | | الكلية الجامعية-ينبع |
| **CSE Department** | | |
| Yanbu Al-Sinaiyah | | ينبع الصناعية |

**BS IN COMPUTER SCIENCE**

**BS IN COMPUTER ENGINEERING**

# Electronics (CSE 252)

**Second Semester - Academic Year 2016– 2017**

**Sem162**

# Ball on Plate Control System

**Supervised by:**

**Ms. Uzma Azeem**

**Members of the group:**

**COURSE: CSE-252** **SECTION: 1**


**1) NAME: Nida Husain Hasan Mukthar. ID NO: 3500903**

**2) NAME: Amal Ali Aleisa. ID NO: 3501026**

**Index:**


| Sr.No | Topic | Page No. |
|---|---|---|
| 1 | Objective | 3 |
| 2 | Introduction | 3 |
| 3 | Hardware Components Used | 4<br>5<br>6<br>7 |
| 4 | Diagrams | 8 |
| 4.1 | Circuit Diagram | 8 |
| 4.2 | Snapshot of the Original designed circuit. | 8 |
| 5 | Detailed Description of the Project | 9 |
| 6 | Security Requirements | 10 |
| 7 | Software Requirements | 10 |
| 8 | Keywords Used | 10 |
| 9 | Conclusion | 10 |
| 10 | References | 11<br>12<br>13 |

## 1. Objective:

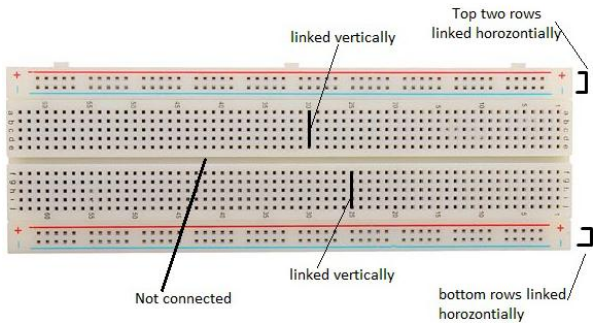
The initial objective of the project is to maintain a static ball position on the plate, rejecting position disturbances. An extension of this objective is movement to specified positions on the system: given a position, a trajectory is plotted and the ball is moved to the new location.

## 2. Introduction:

This project is used to develop the basic understanding of control system. The system to balance a ball could be replaced by other object also in the robotic transportation applications and in simulators. The goal of this project is to develop a ball-on-plate balancing system, capable of controlling the position of a ball on a plate. We intend that the initially horizontal plate will be tilted along each of two horizontal axes in order to control the position of the ball. Each tilting axis will be operated on by an electric motor. Each motor will be controlled using software, with a minimum of position feedback for control.

# 3. Hardware Components Used:

## *a. Bread board.*




A breadboard which is used for testing prototype projects which has pins to insert components into it. The main benefit of it is that it can be reused for other projects. The holes in rows are linked horizontally, and column holes are linked vertically. Almost all people start to learn about electronics on this breadboard.

## *b. Jumper wires.*

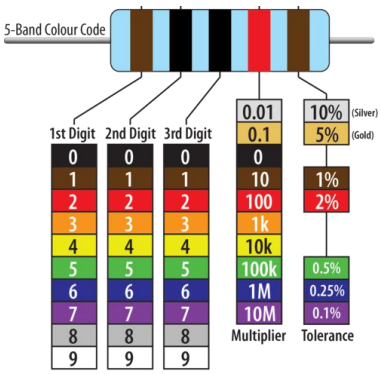
A jumper wire is a conducting wire used to transfer electrical signals between two points in a circuit. The wires can either be used to modify circuits or to diagnose problems within a circuit

## *c. 9 V battery with its clip.*

A battery is an electrochemical cell (an enclosed and protected material) that can be charged electrically to provide a static potential for power or released electrical charge when needed.

## *d. Two 330 ohm resistors*

A resistor is an electrical component that limits or regulates the flow of electrical current in an electronic circuit. It is a two-terminal device that has a fixed relationship between the current passing through it, and the voltage drop across it. This relationship is described in Ohm's law, which states that "the strength of a direct current is directly proportional to the potential difference and inversely proportional to the resistance of the circuit." It does not have any polarity.



## *e. Two LDR (photo-resistor).*

An LDR is a passive electronic component, basically a resistor which has a resistance that varies depending on the light intensity. It does not have a polarity. An LDR or photo-resistor is made from a semiconductor material with a high resistance. When it is dark, it has a high resistance. Since there are very few electrons that are free and able to move, the vast majority of the electrons are locked into the crystal lattice and unable to move. Therefore in this state there is a high LDR resistance. As light falls on the semiconductor, the light photons are absorbed by its lattice and some of their energy is transferred to the electrons. This gives some energy to break free from the crystal lattice so that they can then conduct electricity. This results in a lowering the resistance of the semiconductor and hence the overall LDR resistance. The process is progressive, and as more light shines on the LDR semiconductor, so more electrons are released to conduct electricity and the resistance falls further.

## *f. Joystick module*

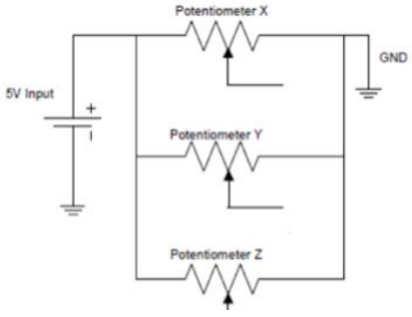
It is made by mounting two potentiometers at a 90 degrees angle. The potentiometers are connected to a short stick centered by springs. This module produces an output of around 2.5V from X and Y when it is in resting position. Moving the joystick will cause the output to vary from 0V to 5V depending on its direction.

**Specifications:**

- Directional movements are simply two potentiometers - one for each axis
- Compatible with Arduino interface
- The biaxial XY Joystick Module KY-023 applies ARDUINO
- Dimensions: 1.57 in x 1.02 in x 1.26 in (4.0 cm x 2.6 cm x 3.2 cm)

**Pin Configuration:**

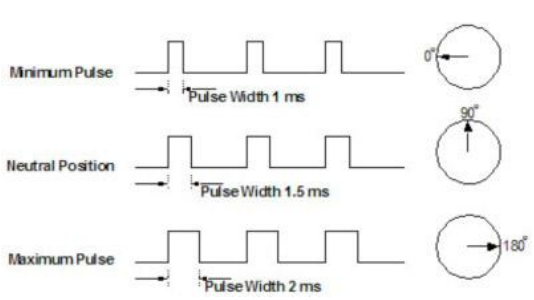
1. GND: ground
2. +5V: 5V DC
3. VRx: voltage proportional to x position
4. VRy: voltage proportional to y position
5. SW: switch pushbutton

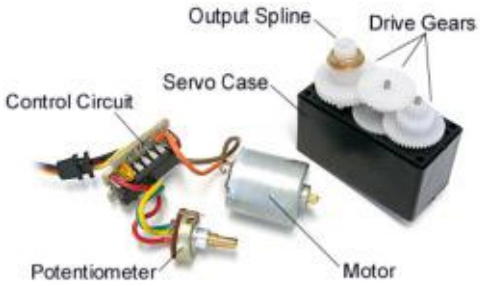


## *g. 3-axis accelerometer/gyroscope (MPU 6050)*


IMU sensors usually consist of two or more parts. Listing them by priority, they are the accelerometer, gyroscope, magnetometer, and altimeter. The MPU 6050 is a 6 DOF (Degrees of Freedom) or a six-axis IMU sensor, which means that it gives six values as output. Three values from the accelerometer and three from the gyroscope. The MPU 6050 is a sensor based on MEMS (Micro Electro Mechanical Systems) technology. Both the accelerometer and the gyroscope are embedded inside a single chip. This chip uses I2C (Inter-Integrated Circuit) protocol for communication. An accelerometer works on the principle of the piezoelectric effect.

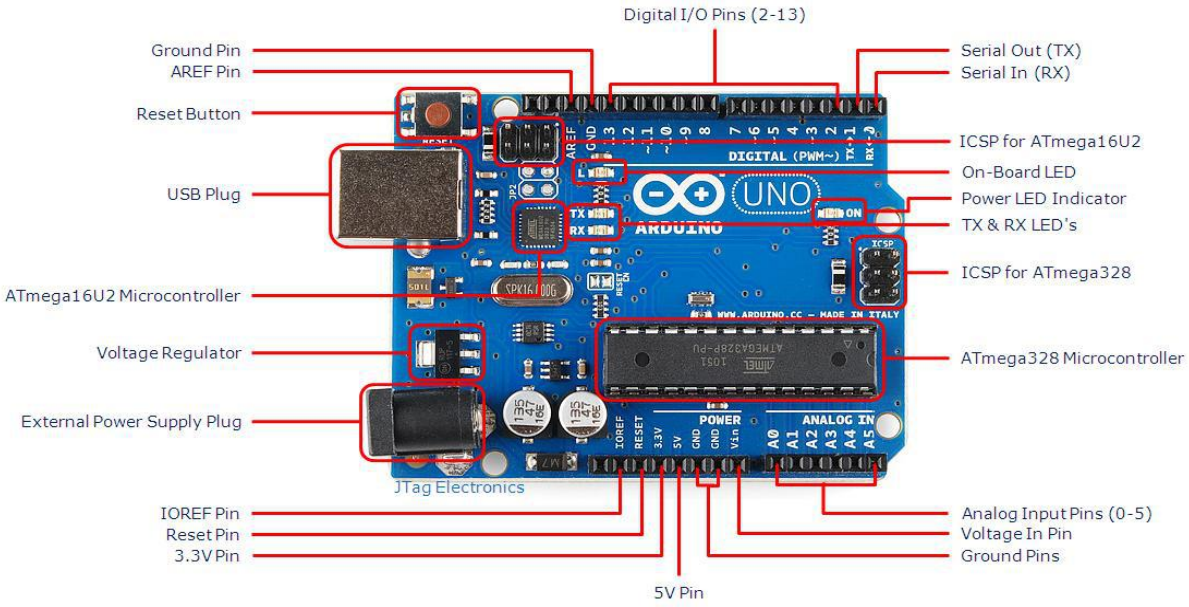
## *h. Servo motor*




A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors. Servomotors is often used to refer to a motor suitable for use in a closed-loop control system.Servos are controlled by sending an electrical pulse of variable width, or **pulse width modulation** (PWM), through the control wire. There is a minimum pulse, a maximum pulse, and a repetition rate. A servo motor can usually only turn 90° in either direction for a total of 180° movement

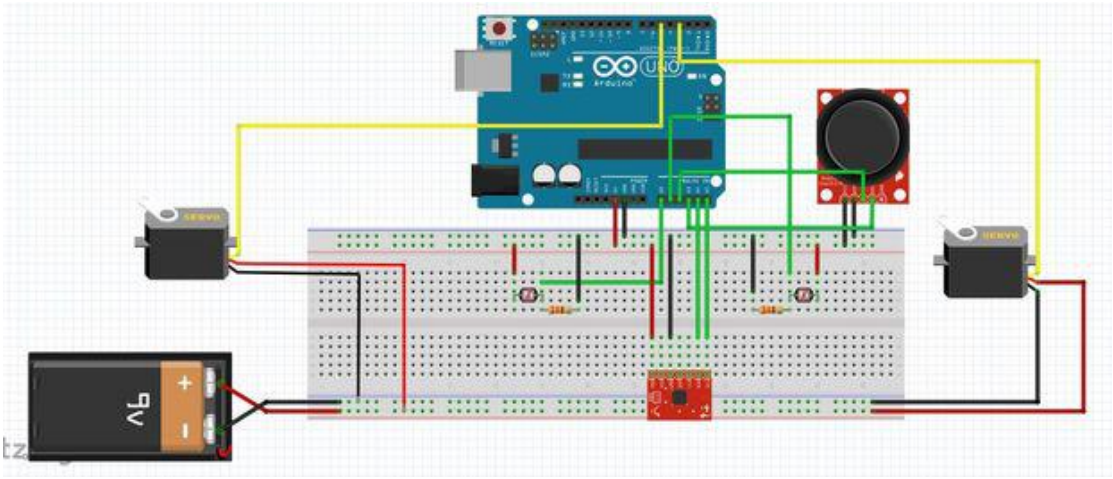
## *i. Arduino Uno*



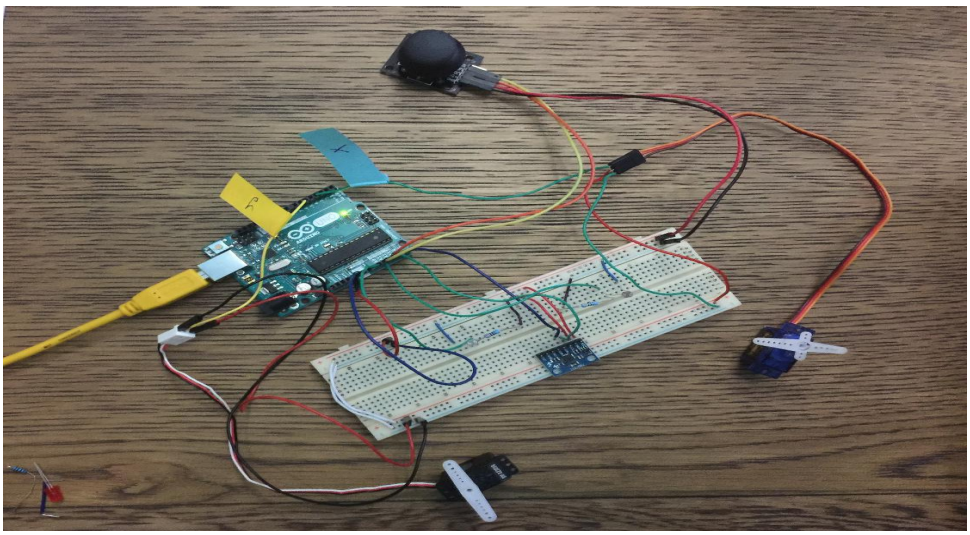

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter. "Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduno, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform.

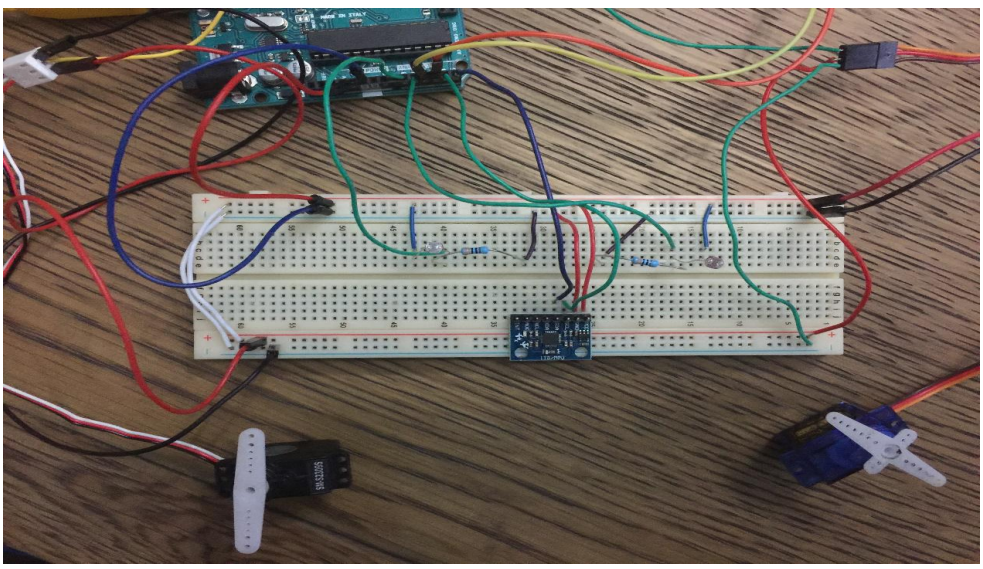
# 4. Diagrams:

## *4.1 Circuit diagram*




## *4.2 Snapshot of your Original designed circuit.*





## 5. Detailed Description of your Project:

The System uses an Arduino as a micro-controller to make a plate that can be controlled using servos. There are 3 operational modes for 0 - 2 number number of users. First of all,The operational mode for 0 users is letting the plate balance itself no matter how you move the box, the plate will always be horizontal! Which could be a great feature when implementing a self-balanced cup holder in the car. Secondly, the remaining modes, mode for 1 and 2, can be implemented in such a way that the players are able to use this system as a game to balance the ball on the plate.The mode for 1 player is using the joystick to keep the ball on the plate. The last mode is similar to the previous one; however, two players can use flash lights to move the plate where both must be playing the game at the same time directing the light on the system.

The Arduino takes signals from the sensors; joystick,accelerometer and photosensor then computes the pwm signals to actuators servo motors. The servo motors control the plate orientation.

The actual signal from the accelerometer is compared against the desired value and multiplied by a constant proportional factor to move the servos.

The same proportional control system is applied for the photocells.

The joystick input signal to the Arduino is mapped to output a suitable PWM signal.

The wiring of the project is as the circuit diagram. The photocells are wired in a voltage divider circuit with the 330 ohm resistor to pins A0 and A1 in the Arduino. The accelerometer is wired to pins A4 and A5. The joystick is connected to pins A2 and A3.The two servos are connected to PWM digital pins 3 and 5. It's supplied by a 9v battery.

The code that is uploaded to the Arduino is split into two tabs, the main tab and the functions tab. The main tab is basically a switch case in the loop function. It calls for the mode checking function and goes to the right case based on what it gets back from that mode checking function.

The functions tab contains 5 functions:

**setplane();** // this functions sets the plane horizontal in the setup function

**checkbuttons();** // this functions checks to see which mode are we in. If the joystick is not in its normal position, it returns that appropriate case number. Similarly if it detects a light value greater than the specified value, its returns back a different case number. If nothing is detected, the default case will be activated.

**followlight();** // keeps a specific distance between the photocell and the flashlight using a proportional control system on the error between desired and actual light value.

**balance();** // does the same as followlight() but uses the accelerometer values instead of light values.

**followjoystick();** // maps the input signal from joystick to the output pwm signal to servo motors.

## 6. Security Requirements:

- It is essential for all the pins to be connected correctly.
- The ground and the power supply should be connected properly. Reversing them can result in components burning out. Most components are not designed to function if the polarity is reversed. Such a mistake can cause a short circuit and can result in smoke.
- It is necessary to make sure the surface on which the system is placed is free of water, and other such materials that can damage the machine. This is due to the fact that water is a conductor of electricity and can cause a short circuit.

## 7. Software Requirement:

- ARDUINO 1.8.2
  *The open-source Arduino Software (IDE)*

- Microsoft Word
  *Word processing software that allows users to create and edit text documents.*

## 8. Keywords Used:

**Piezoelectric Effect** is the ability of certain materials to generate an electric charge in response to applied mechanical stress.

**PWM** Pulse Width Modulation, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off.

## 9. Conclusion:

In conclusion, overall the system can be considered a success. The objective of the project was to control the position of the ball on the plate for static positions and smooth paths, as well as have the system capable of correcting for disturbances in ball position. In general these goals were accomplished, especially static position balancing and disturbance rejection. The ball can be centered on the plate or another coordinate location, and will quickly correct for even large disturbances to the ball position. Furthermore, this system can be implemented in various ways. First of all, as a game where the players can balance the ball on the plate either by using the joystick, when there is a singular player, or using flash lights when there are two players using this system. Otherwise, the system when balancing itself without external forces or players can be implemented as a self-balanced cup holder in cars. All in all, this system was successfully designed and has great potential to be implemented into various applications.

# 10. References:

## a. https://diyhacking.com/arduino-mpu-6050-imu-sensor-tutorial/

## b. Code

**Main tab :**

```
#include <Servo.h>
#include<Wire.h>

int initial_photo1;   //defining integers
int initial_photo2;
int initial_photo;
int desired1;
int desired2;
int button;

Servo servox;        // defining objects
Servo servoy;

const int MPU = 0x68; // I2C address of the MPU-6050
int16_t AcX, AcY;


void setup() {

  initial_photo1 = analogRead(A0);   // setting initial values for automatic calibration
  desired1 = initial_photo1 +100;
  initial_photo2 = analogRead(A1);
  desired2 = initial_photo2 +100;
  initial_photo = max(initial_photo1, initial_photo2);  // comparing values and storing the maximum
  button = checkbuttons();  // check which mode are in

  servox.attach(5);  // attaching servos
  servoy.attach(3);
  setplane();   // initial setting plane to horizontal
  Serial.begin(9600);  // initiating serial communication

  Wire.begin();  // begin I2C communication
  Wire.beginTransmission(MPU); //begin transmission of bytes
  Wire.write(0x6B);  // PWR_MGMT_1 register
  Wire.write(0);     // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true); // end transmission of bytes


}

void loop() {
  checkbuttons();  // check for mode

  switch (button) {
    case 1:

      while ((analogRead(A0) > initial_photo + 5) && (analogRead(A1) > initial_photo + 5))  // stay in this loop
until the light goes back to initial
      {
```

```
      followlight(); // follow light program
     }

     break;

   case 2:
     while ((analogRead(A2) / 8 != 63)  || (analogRead(A3) / 8 != 63))  // while joystick is not in zero position
     {
      followjoystick();  // follow the joystick program
     }
     break;

   default:

     balance(); // Keep balancing the plate as a default mode

     break;

  }
}
```

## Function tab:

```
void setplane()
{
 servox.write(100);
 servoy.write(100);

}
int checkbuttons()
{

 if ((analogRead(A0) > desired1)  && (analogRead(A1) > desired2))       // if lights are detected
 {
  button = 1; // go to mode 1

 }
 else if ((analogRead(A2) / 8 != 63)  || (analogRead(A3) / 8 != 63))  // if joystick is detected
 {
  button = 2;// go to mode 2
 }
 else button = 0; // otherwise keep balancing the plate

 return button; // return mode
}
void followlight()  //might need more effiency
{

 double diffx = (desired2) - (analogRead(A1));  // error in x direction
 double diffy = (desired1) - (analogRead(A0));  // error in y direction
 double prop = 0.1;  // proportional control factor
 if ((abs(diffx) > 20) && (servox.read() >= 60 ) && (servox.read() <= 140))
 {
  servox.write(servox.read() + (diffx * prop)); // move servo on x direction by proportional value
 }

 if ((abs(diffy) > 20) && (servoy.read() >= 60 ) && (servoy.read() <= 140))
 {
  servoy.write(servoy.read() - (diffy * prop)); // move servo on y direction by proportional value
 }

 // keep servos within the range
 if (servox.read() > 140) {
  servox.write(140);
 }
 if (servox.read() < 60) {
```

```
  servox.write(60);
 }
 if (servoy.read() > 140) {
  servoy.write(140);
 }
 if (servoy.read() < 60) {
  servoy.write(60);
 }
delay(10);
}
void balance()
{

 Wire.beginTransmission(MPU);
 Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
 Wire.endTransmission(false);
 Wire.requestFrom(MPU, 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)
 AcX = AcX - 1500; // setting accelerometer value horizontal at 0
 AcY = AcY + 2500; // setting accelerometer value horizontal at 0



 double diffx = 0 - AcX / 100; // error in x direction
 double diffy = 0 - AcY / 100; // error in y direction
 double prop = 0.15; // proportional control factor
 if ((abs(diffx) > 5) && (servox.read() <= 140) && (servox.read() >= 60))
 {
  servox.write(servox.read() + (diffx * prop)); // move servo on x direction by proportional value
 }

 if ((abs(diffy) > 5) && (servoy.read() <= 140) && (servox.read() >= 60))
 {
  servoy.write(servoy.read() - (diffy * prop)); // move servo on y direction by proportional value
 }

 // keep servos within the range
 if (servox.read() > 140) {
  servox.write(140);
 }
 if (servox.read() < 60) {
  servox.write(60);
 }
 if (servoy.read() > 140) {
  servoy.write(140);
 }
 if (servoy.read() < 60) {
  servoy.write(60);
 }

delay(25);
}
void followjoystick()
{
 int  joystickx = map(analogRead(A3), 0, 1023, 140, 60); // map joystick x values
 int  joysticky = map(analogRead(A2), 0, 1023, 60, 140); // map joystick y values
 servox.write(joystickx);// move servo x
 servoy.write(joysticky);// move servo y

}
```

*****END*****