# Applying Classical Control Theory to an Airplane Flap Model on Real Physical Hardware

Edgar Collado Alvarez, BSIE, EIT, Janice Valentin,
Jonathan Holguino ME.ME in Design and Controls
*Polytechnic University of Puerto Rico, ecollado@pupr.edu,, janicenvt@gmail.com, jholguino@pupr.edu*

*Mentor*: Prof. Bernardo Restrepo, Ph.D, PE in Aerospace and Controls
*Polytechnic University of Puerto Rico Hato Rey Campus, brestrepo@pupr.edu*

***Abstract* — For the trimester of Winter -14 at Polytechnic University of Puerto Rico, the course Applied Control Systems was offered to students interested in further development and integration in the area of Control Engineering. Students were required to develop a control system utilizing a microprocessor as the controlling hardware. An Arduino Mega board was chosen as the microprocessor for the job. Integration of different mechanical, electrical, and electromechanical parts such as potentiometers, sensors, servo motors, etc., with the microprocessor were the main focus in developing the physical model to be controlled. This paper details the development of one of these control system projects applied to an airplane flap.**

***After physically constructing and integrating additional parts, control was programmed in Simulink environment and further downloads to the microprocessor. MATLAB's System Identification Tool was used to model the physical system's transfer function based on input and output data from the physical plant. Tested control types utilized for system performance improvement were proportional (P) and proportional-integrative (PI) control.***

***Key Words* — embedded control systems, proportional integrative (PI) control, MATLAB, Simulink, Arduino.**

## I. INTRODUCTION

During the master's course of Mechanical and Aerospace Control System, students were interested in developing real-world models with physical components and parts to be controlled and analyzed with tools and concepts discussed in class. Therefore, a special topic course was offered for that purpose as a continuation of what was already learned. As it is known, theory can slightly be different from what is observed in a real-life application: non-linearities, noise, windup, constraints, and unpredictable perturbations (behavior) are always present and observed when controllers are tested against a reacting system.

For this course, students developed components and integrated physical electromechanical parts to set up models in which control was embedded by use of the Arduino Mega microprocessor. Our experimentation, integration, and learning process have been more complex than expected and documented in this paper. The real hardware experience was tested in an airplane flap and it is detailed in the following sections.

## II. FIRST PHASE - POTENTIOMETER AND SERVO MOTOR CONTROL

The team started in the development of a potentiometer and servo motor control circuit. The purpose is to be able to apply it to the control and movement of an airplane flap. A circuit consisting of 2 potentiometers and 2 servomotors can be observed below.

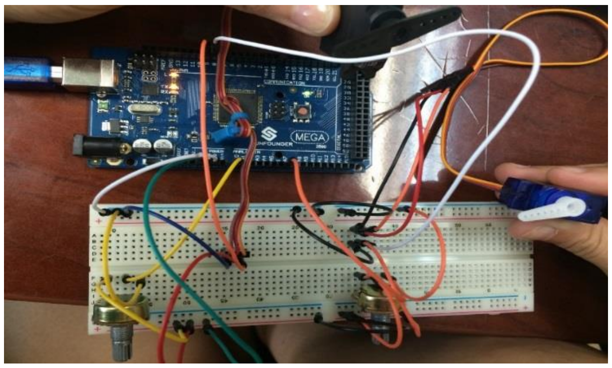


Figure 1. Two potentiometer-two servomotor control circuit physical connection.

After physical connection is complete one must configure the Arduino compiler to the settings needed to give the code instructions to the selected board. In this case, board selection is achieved by choosing *Tools> Board> Arduino Mega2560 or Mega ADK* as shown in figure 2.

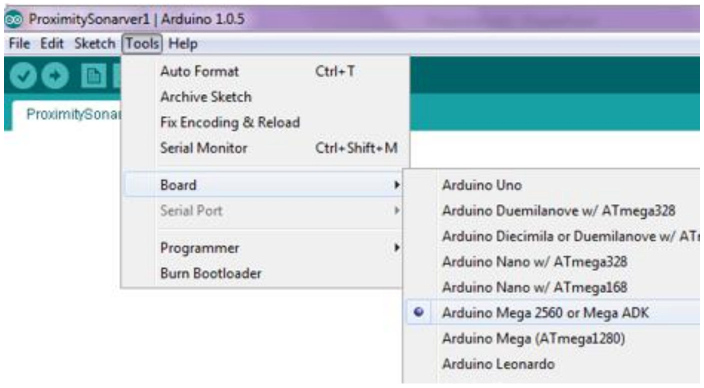

ProximitySonarver1 | Arduino 1.0.5
File Edit Sketch Tools Help
ProximitySonar
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board
Serial Port
Programmer
Burn Bootloader
Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATr
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo

Figure 2. Selection of Arduino board type.

Then, COM port where the Arduino board communicates with the computer must be selected. This is achieved by selecting *Tools> Serial port* and choosing the available COM port. Arduino C++ code for this circuit can be found in the appendix section as **_2servos2potsprogram.ino**. After this, a second circuit was mounted where 2 potentiometers and 1 servo motors were involved. Physical Setup is shown in Figure 3.

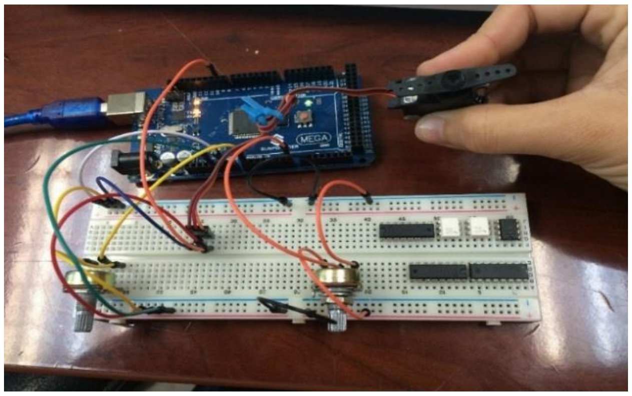


Figure 3. Two potentiometer, one servomotor control circuit physical connection.

This type of circuit is ideal for our wing span control application. Therefore, a schematic of this was developed using Fritzing Arduino and electronic board software. This is observed below:

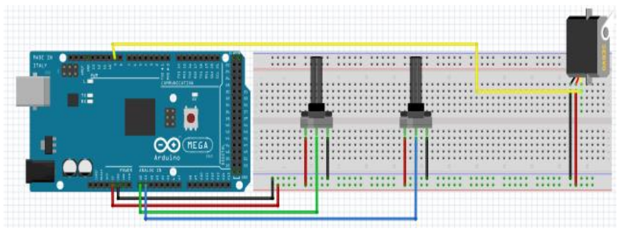


Figure 4. Fritzig software sketch for one potentiometer, one servomotor control circuit.

Arduino code for this application can be found in the appendix section as **_1servos2potsprogram.ino**.

These codes were tested to verify that the signals from the potentiometers and the servomotors were communicating effectively with the Arduino microprocessor. The team decided as a next task to implement this code's logic in the Simulink environment because the block diagram approach in control theory is better for teaching and learning.

## III. PHASE II - POTENTIOMENTER AND SERVO MOTOR CONTROL

A 3D printed model of an airplane flap was constructed using the Department of Mechanical Engineering's very own 3D printer. The conceptual design of this wing was done using **CREO** modeling software. Individual parts were drawn and later assembled as shown in Figure 5. \

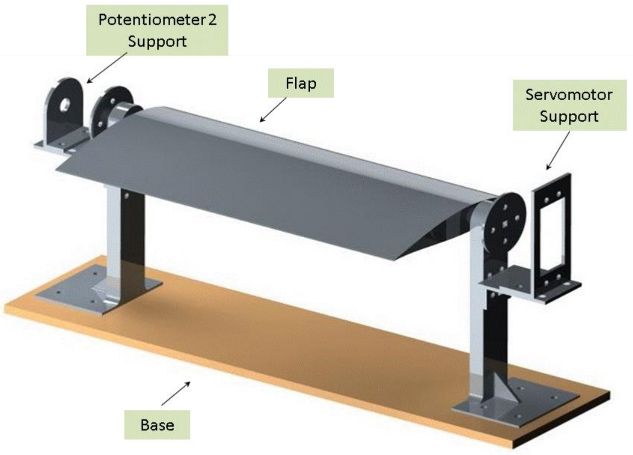
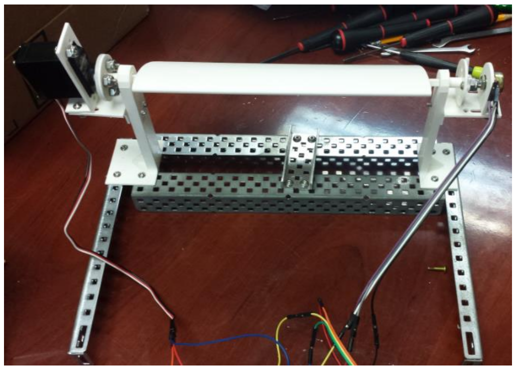
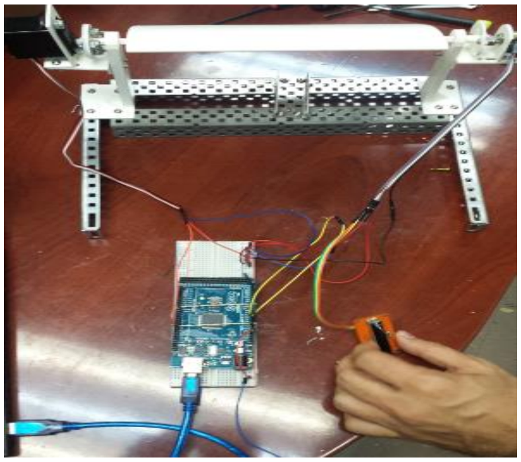


Figure 5. CREO modeling software sketch for airplane wing instructional model.

The physical construction of the model can be observed in Figure 6(a). A potentiometer is located on one side which acts as the feedback sensor of the flap angle. At the other side, the servo motor is the actuator responsible for minimizing the difference between the desired and actual flap angle. In Figure 6(b), it can be observed the physical flap integrated with the control circuit.



(a)

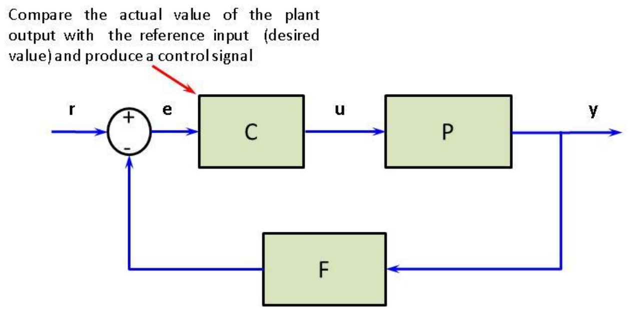


(b)

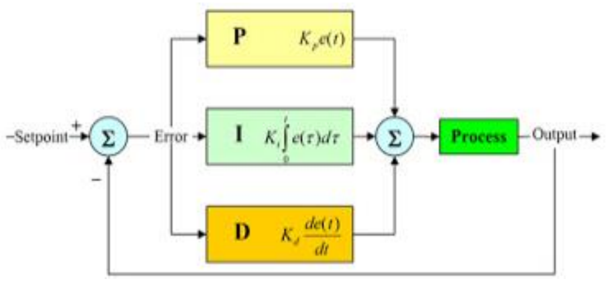
Figure 6. Physical construction of the airplane flap model. Servomotor and potentiometer attached to the flap are appreciated. Connection of these with Arduino board is also shown (bottom).

## IV. PHASE III – TWO POTENTIOMETER AND SERVOMOTOR CONTROL

From class theory [1], the team learned that a closed loop control system is modeled by the following block diagrams.


Compare the actual value of the plant output with the reference input (desired value) and produce a control signal
r
e
C
u
P
y
+
-
F

(a)



(b)

Figure 7. Model of a controlled closed loop feedback system block diagram (a). Model of a closed loop feedback system block diagram with PID control (b) PID flow is detailed.

The smaller case letters in Figure 7(a) represent the following:

- r – input signal or setpoint.
- e – error.
- u – output signal of the controller, input to the plant.
- y – output or system response to input.

The upper case letters represent the following:

- C – Controlling element
- P – Plant or model of the system
- F – Feedback element (sensor or in this case, potentiometer2)

The purpose of a PID controller is to add the characteristics of proportional, integral and derivative controller types so that a system or process (the airplane flap) responds to disturbances or changes in the set-point as required or desired. The team concentrated on making fine tuning of the proportional (P), proportional integrative (PI), and proportional integrative derivative (PID) control for this project. See ***Results*** section V for further details.

Final Simulink Diagram for the control of the airplane flap is shown in Figure 8.

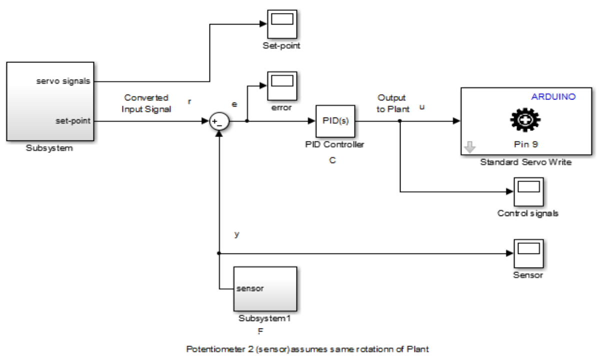


Figure 8. Simulink Block Diagram controlling the airplane wing using Arduino as the acting agent of control.

The input signal from the potentiometer is read by the Arduino. The ***Subsystem*** block in Figure 8 is a custom block. It represents the **conversion** of Arduino read input signal from an encoded digital value to an angle $\theta_{Pot1}$ in degrees. This permits to visualize the physical angular position of the potentiometer. See figure 9 for details of components of this block.

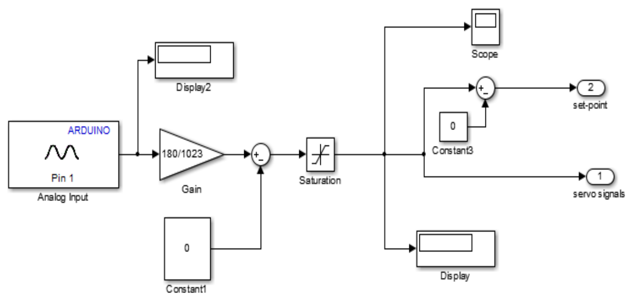


Figure 9. Subsystem block exploded view.

The conversion rate to map the digital input signal with the angle was found to be 180/1023, which gives the value in degrees. The ***saturation*** block in Figure 9 is placed to ensure that the range of values given by the potentiometer is between 0 to 180 degrees. This is a preventive measure due to the possibility of physically turning the knob of the potentiometer to these values with use and abuse of the hardware.

***Subsystem1 (Sensor)*** block in Figure 8 is detailed in Figure 10. It is another custom block that represents the conversion of signal from an encoded digital value read by Arduino to an angle $\theta_{Pot2}$ in degrees. It is worthwhile to mention that this potentiometer is **physically connected** to the airplane flap by a shaft between them. The shaft serves as the feedback element, connecting the servo with the sensor (potentiomenter2) to give signal data to the Arduino for further processing/conversion on Simulink.

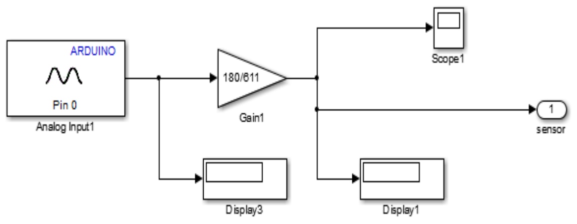


Figure 10. Subsystem1 block exploded view.

The conversion rate was found to be 180/611, because the range of the potentiometer operate by the servomotor (180 degrees) does not complete a full revolution (1023 signal). The final value after the conversion rate is displayed in degrees.

The important variable values gained from the simulation are: set-point (input "r"), error, sensor, and control signal (output "u").

## V. RESULTS

The behavior of the airplane flap with the open loop plant is shown in Figure 11.

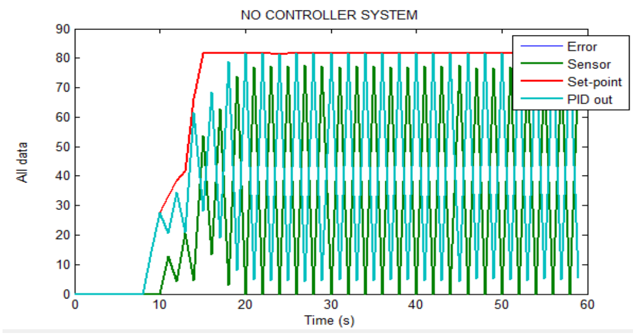


Figure 11. Airplane wing system response for No Control Type.

The set-point value of Potentiometer 1 was placed at 80 degrees. The plant output measured by the sensor is unstable. Therefore, the system is unstable by itself (no controller).

Adding proportional and integral control (I =1) to the system, the system response improved as shown in Figure 12.

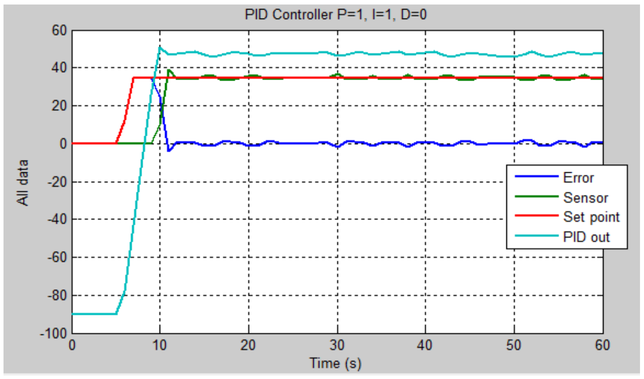


Figure 12. Airplane wing system response for P=1 and I = 1 Control Type.

Set-point value for this case was placed around 35 degrees. After the controller tracks the set point value, the error signal starts to fluctuate around 0. This occurs around the 12 seconds mark and continuous for the rest of the simulation. Also note that the set-point and sensor signals have a difference around 0 around this time. Though not a perfect 0 difference (no error), this may be due to the controller constants are not tune to completely mitigate the oscillation of the interaction of the electromechanical parts.

Increasing proportional constant to 1.5, the system response observed in Figure 13.

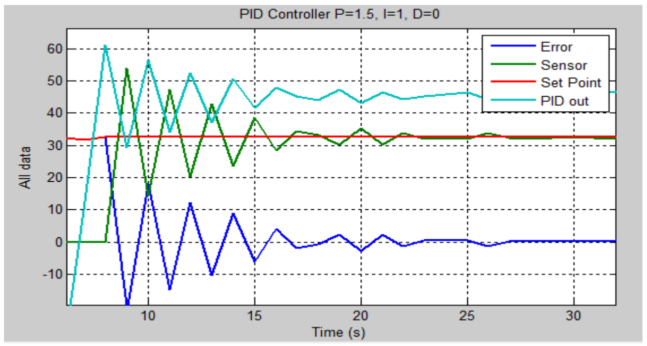


Figure 13. Airplane wing system response for P = 1.5 and I = 1 Control Type.

Set-point value was placed at 32 degrees. Error signal starts to fluctuate around 0 until 28 seconds. Also note that the set-point and sensor signals have a difference around 0 around this time. Overshoot is observed for the sensor signal. Compared to the last simulation, this increase in proportional gain has caused a fast system response (rise time around 6-7 seconds vs. 12 seconds before). In this case the overshoot was around 68% compared to approximately 14% from a before set-point of 35 degrees.

Decreasing proportional constant to 0.1, the system response observed is shown in Figure 14.

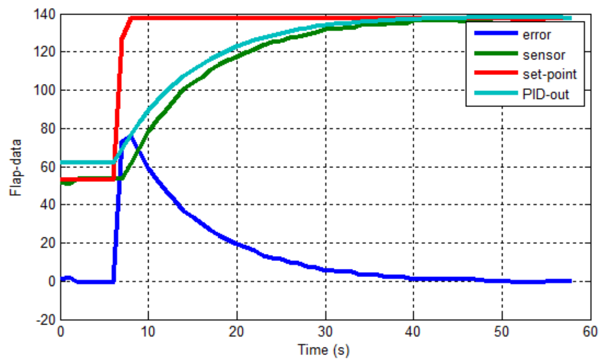


Figure 14. Airplane wing system response for P= 0.1 and I = 1 Control Type.

Set-point value was placed around 138 degrees. The system was sluggish in tracking the set point but all the oscillations were mitigated. Also note that the set-point and sensor signals have a difference around 0 around this time. No overshoot is observed for the sensor signal. This means that the response of the systems is closed to overdamped behavior. Also note that the rise time for this system occurs for when steady state time is achieved (around 50 seconds, too).

## VI. CONCLUSION

The team was excited on all phases of design for applying control system theory to a mechatronic system. The learning experience acquired in this project has been of great value to all of us, as control was achieved using both Arduino C++ code and Simulink commands on a microprocessor. The results related to proportional (P) and proportional integrative (PI) control have been consistent with what was learned in the classroom. Slight variability in system stability around the set-point values were due to the behavior of the integration between electrical and mechanical components, which is imperfect by nature. It is desired for the team to develop a new paper explaining the use of the system identification technique employed in this work. Also, the team looks forward to write another paper where other control strategies such as optimal or adaptive control applied to be explained.

## VII. REFERENCES

[1] Ogata, K., 2002, *Modern Control Engineering*, 4th Ed. Prentice Hall, pp. 219-294

[2] Napolitano M., 2012, Aircraft Dynamic from Modeling to Simulation, John Wiley & Sons.

[3] Ljung, L., 1999, System Identification: Theory for the User, 2nd Ed. Prentice Hall.

[4] Banzi, M. 2011, Getting Started with Arduino, 2nd Ed., Maker Media.

[5] Nuruzzaman, M., 2005, Modeling and Simulation in Simulink for Engineers and Scientific, AuthorHouse.

[6] Kurniawan, A., 2013, Getting Started With Matlab and Arduino, PE Press.

## *Appendix A: Arduino C++ code programs*

### A.1 _2servos2potsprogram.ino

```
#include <Servo.h>
 Servo myservo;  // create servo object to control a servo
Servo myservo2;
 int potpin = 0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin
int potpin2 = 7;  // analog pin used to connect the potentiometer
int val2;    // variable to read the value from the analog pin
void setup()
{
  Serial.begin(9600);
  myservo.attach(9);  // attaches the servo on pin 9 to the servo
object
  myservo2.attach(8);  // attaches the servo on pin 9 to the servo
object
}

void loop()
```

```
{
  Serial.print("PotVal is ");
  Serial.print(val);

  Serial.println("PotVal is ");
  Serial.print(val2);
  val = analogRead(potpin);            // reads the value of the
potentiometer (value between 0 and 1023)
  val = map(val, 0, 1023, 0, 180);     // scale it to use it with the
servo (value between 0 and 180)
  myservo.write(val);                  // sets the servo position according
to the scaled value
  delay(15);   // waits for the servo to get there
  val2 = analogRead(potpin2);            // reads the value of the
potentiometer (value between 0 and 1023)
  val2 = map(val2, 0, 1023, 0, 180);     // scale it to use it with the
servo (value between 0 and 180)
  myservo2.write(val2);                  // sets the servo position
according to the scaled value
  delay(15);
}
```

## A.2 _1servos2potsprogram.ino

```
int error=0;
int angulo=0;
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
//Servo myservo2;
 int potpin = 0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin
int potpin2 = 7;  // analog pin used to connect the potentiometer
int val2;    // variable to read the value from the analog pin
void setup()
{
  Serial.begin(9600);
  myservo.attach(9);  // attaches the servo on pin 9 to the servo
object
 //myservo2.attach(8);  // attaches the servo on pin 9 to the servo
object
}
 void loop()
{
   val = analogRead(potpin);            // reads the value of the
potentiometer (value between 0 and 1023)
  val = map(val, 0, 1023, 0, 180);     // scale it to use it with the
servo (value between 0 and 180)
  //myservo.write(val);                  // sets the servo position
according to the scaled value
  //delay(5);   // waits for the servo to get there
    val2 = analogRead(potpin2);            // reads the value of the
potentiometer (value between 0 and 1023)
  val2 = map(val2, 0, 1023, 0, 180);     // scale it to use it with the
servo (value between 0 and 180)
  //myservo2.write(val2);                  // sets the servo position
according to the scaled value
  //delay(15);
    error = val2 -val;
  angulo=error;
  angulo=map(angulo,-180, 180, 0,180);
   myservo.write(angulo);
   delay(20);

   Serial.print("Val es        \t");
  Serial.println(val);
   Serial.print("Val2 es              \t");
  Serial.println(val2);
   Serial.print("El error:                      \t ");
  Serial.println(error);
  Serial.print("El angulo:                              \t ");
  Serial.println(angulo);
}
```