```
// Amateur Radio Satellite Tracker
// by Glen Popiel, KW5GP

/*

 This program is free software: you can redistribute it and/or modify
 it under the terms of the version 3 GNU General Public License as
 published by the Free Software Foundation.

 This program is distributed in the hope that it will be useful,
 but WITHOUT ANY WARRANTY; without even the implied warranty of
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 GNU General Public License for more details.

 You should have received a copy of the GNU General Public License
 along with this program.  If not, see <http://www.gnu.org/licenses/>.

 */

#include <Servo.h>  // Use the Servo Library
#include <Wire.h>  // Use the I2C Communication Library
#include <LiquidCrystal_I2C.h>  // Use the LiquidCrystal_I2C Library

LiquidCrystal_I2C lcd(0x27,16,2);  // set the LCD address to 0x27 for a
16 chars and 2 line display

Servo myservoAZ;  // create servo object to control Azimuth servo
Servo myservoEL;  // create servo object to control Elevation servo

int min_servo_pulse = 900;  // Set the Azimuth Servo pulse width for 0
Degrees
int max_servo_pulse = 1970;  // Set the Azimuth Servo pulse width for 360
Degrees
int EL_Min_pulse = 640;  // Set the Elevation Servo pulse width for 0
Degrees
int EL_Max_pulse = 2340;  // Set the Azimuth Servo pulse width for 180
Degrees
int delay1 = 50;
int delay2 = 5000;
int currentEL = 0;  // Variable to hold the current elevation
int currentAZ = 0;  // variable to hold the current azimuth
int inByte;  // the incoming byte on the Serial port
boolean move_command = false;  // Flag to indicate we are building a move
command
int byte_count = 0;  // the number of bytes received on the Serial port
char az_buffer[4];  // Holds the decoded azimuth command
char el_buffer[4];  //  Holds the decoded elevation command
char rotor_buffer[10];  // The incoming Rotor command data
int set_az, set_el;  // Holds the desired azimuth and elevation data

void setup()
{
  lcd.init(); // initialize the lcd
  lcd.backlight();  // Turn on the LCD backlight
```

```
  lcd.setCursor(0,1);
  lcd.print("KW5GP SatTrack");
  lcd.setCursor(0, 0);
  lcd.print("Rotor Controller");
  delay(delay2);
  myservoAZ.attach(9);  // attaches the servo on pin 9 to the Azimuth
servo object
  myservoEL.attach(10);  // attaches the servo on pin 10 to the Elevation
servo object
  pinMode(13, OUTPUT);   // initialize the pin 13 as an output.
  Serial.begin(9600);  // Start the Serial Port at 9600 baud
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Homing Servos     ");
  set_azimuth(0);  // Set the Servos to 0 degrees
  set_elevation(0);
  lcd.setCursor(0,1);
  lcd.print("Elevation:  0");  // Display the current Elevation
  lcd.setCursor(0, 0);
  lcd.print("Azimuth  :  0");  // Display the current Azimuth
  delay(delay2);
}

void loop()
{
  if (Serial.available())     // read data from the Serial Port
  {
    inByte = Serial.read();
    Serial.write(inByte);

    if (inByte == 87) // If the incoming data is a "W", it's a rotor move
command
    {
      // Turn on string build - next 7 chars = az and el
      move_command = true;
      byte_count = 0;
    }
    //  If it's not a move command - add to string if we're building
    if (move_command and inByte !=87 )
    {
      rotor_buffer[byte_count] = char(inByte);
      byte_count = byte_count + 1;
      if (byte_count <= 3)  // The first 3 characters are Asimuth
      {
        az_buffer[byte_count - 1] = char(inByte);
      }
      if (byte_count >= 5 and byte_count <= 7)  // The last 3 characters
are Elevation
      {
        el_buffer[byte_count -5] = char(inByte);
      }
      if (byte_count == 8) {  // Once we have 8 characters, we have a
complete move command
```

```
        set_az = atoi(az_buffer);  // Convert the Azimuth data to an
integer
        set_el = atoi(el_buffer);  // Convert the Elevation data to an
integer
        set_azimuth(set_az);  // Move the Azimuth Servo to desired
position
        lcd.setCursor(11,0);
        lcd.print(set_az);  // Display the New Azimuth
        lcd.print("   ");
        set_elevation(set_el);  // Move the Elevation Servo to desired
position
        lcd.setCursor(11,1);
        lcd.print(set_el);  // Display the New Elevation
        lcd.print("   ");
        move_command = false;  // Indicate we're done with this move
command
      }
    }
  }
}

// ---------------------------------------------------------------------
--------------------
// Functions

// Set Azimuth Funtion
void set_azimuth(int desired_az)  // Moves the Azimuth Servo to desired
position
{
  int move_azimuth;
  move_azimuth = map(desired_az,360, 0,min_servo_pulse,max_servo_pulse);
// Map the desired position to the correct Azimuth servo pulse width
  myservoAZ.writeMicroseconds(move_azimuth);  // Send the Servo position
pulse to the Azimuth servo
  delay(delay1);  // Wait for move to complete
}

// Set Elevation Function
void set_elevation(int desired_el)  // Moves the Elevation Servo to
desired position
{
 int move_elevation;
 move_elevation = map(desired_el,0, 180,EL_Min_pulse,EL_Max_pulse);  //
Map the desired position to the correct Elevation servo pulse width
 myservoEL.writeMicroseconds(move_elevation);  // Send the Servo position
pulse to the Elevation servo
 delay(delay1);  // Wait for move to complete
}
```