B EE 526: Advanced Topics in Embedded Systems Design

2 Axis Solar Tracker

José Rubén García Villalobos

Professor: Arnie Berger, Ph.D.

José Ruben Garcia
BEE 526 Adv. Topics on Embedded Systems Design

Contents

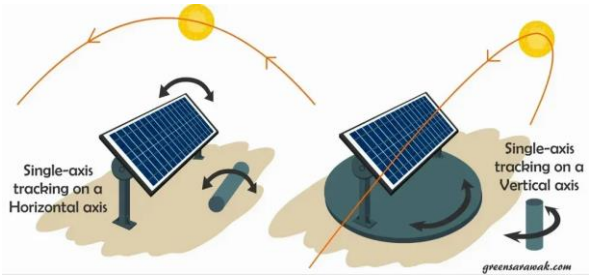
José Ruben Garcia
BEE 526 Adv. Topics on Embedded Systems Design

2-axis Solar Tracker Project

Design goals

The main design goal for this project is to build a 2-axis light follower, or tracker. It can not only follow the sun at a slow rate, but also follow any source of light at real time and adjust the position of the base to always face it.

Using 3-5 Light Dependent Resistors connected to a processor, I will control 2 SG90 servo motors to move a base and make it follow the light, one servo motor per axis. The difference in resistance caused by the light source on the LDRs will change the voltage received by the ADC on the board, making it possible to detect where the light source is, and will give the instruction to the servomotors to move in the desired direction. The servomotors have the limitation of 180° movement each, which will limit the movement of the base; but for solar applications there is no need for more than that movement.



After my main goals are completed, there will be a few secondary goals that will be implemented if there is enough time to achieve them.

Starting with our secondary features, an array of solar cells will be mounted and connected to the moving tracker, then, the power output will be measured and compared with the solar tracker active and when it is idle.

Afterwards, a battery storage will be added, putting some rechargeable batteries connected to the solar panels and a charge control unit that will control the charge and discharge of the battery. Also, an output LED array will be connected to demonstrate the harvested energy from the solar array.

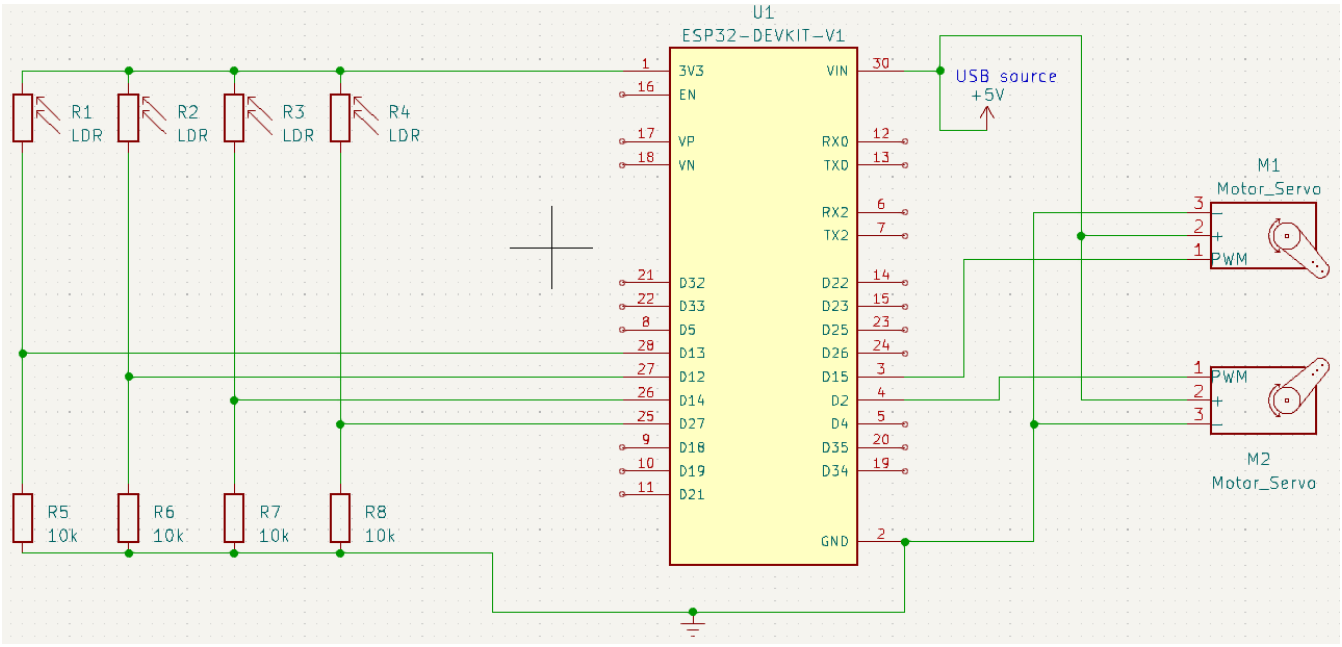
Lastly, a flower-like design will be constructed, where a LDR will act as the primary or starting parameter and will indicate a motor to start running and open the flower-like design.

José Ruben Garcia
BEE 526 Adv. Topics on Embedded Systems Design

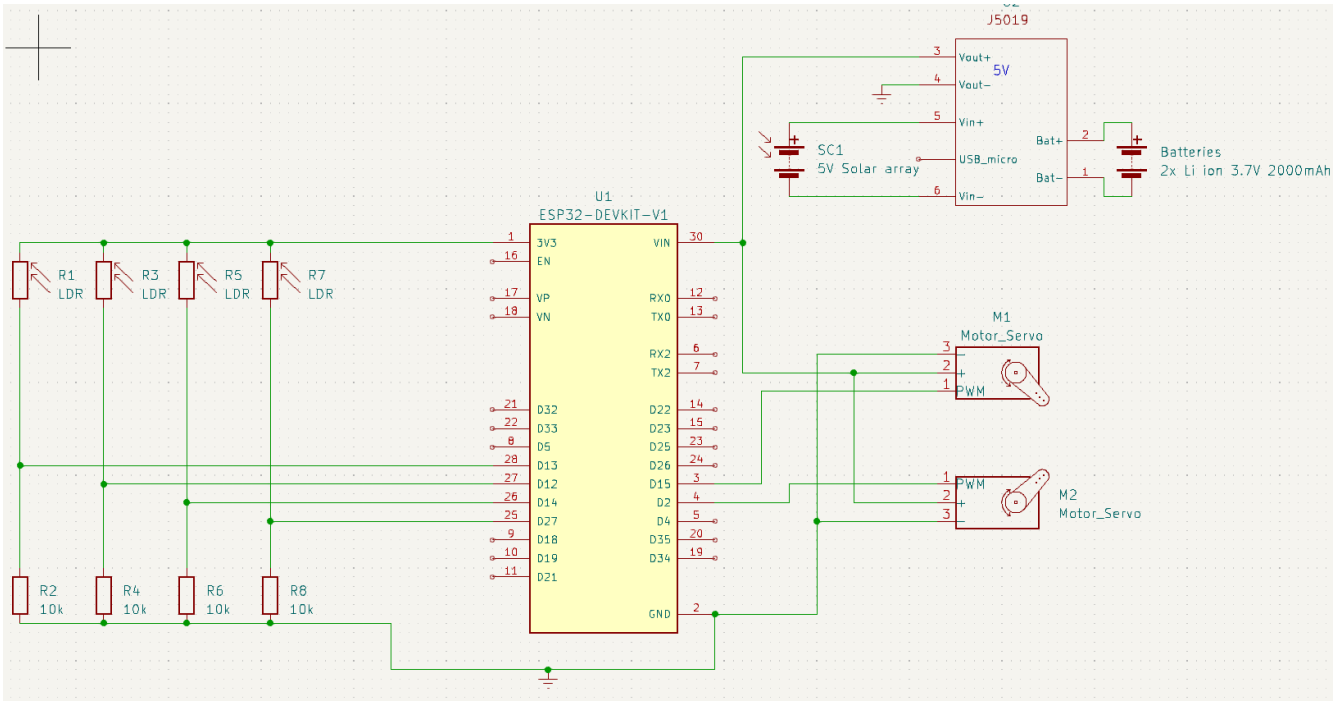
Another feature to be implemented will be the simultaneous comparison and action of the 2 servo motors, using a RTOS on a ESP32 board, creating different tasks that will run alongside on the on-board cores.

The next section on schematics shows 2 different sources, as the second one shows the connections if the secondary feature with the battery charger is implemented.

Schematics



1. Schematics with USB as 5V source



2. Schematics with solar array, 3.7V batteries and boost converter to 5V

José Ruben Garcia
BEE 526 Adv. Topics on Embedded Systems Design

Processor choice

After a little research I changed my original board choice to fit at least 4 ADC inputs and will be using a ESP32 development board to achieve my goals. The ESP32 is much more powerful board than the Raspberry Pi I was intending to use and will perform even better for light source fast tracking. Another additional feature is that this board has 2 cores and enough processing power to run tasks simultaneously, which will give me the ability to pin each of the processing of the 2 servo motors to the individual 2 cores on the ESP32.

Motivation for this project

I have always liked and have been interested in renewable energy applications and efficiency. This project sums up this interest, combined with embedded systems application.

I enjoy making projects that can be physically implemented, as you get more contact with what you are doing and can see results first hand. The combination of a mechanic and electric project really makes me eager to do it.

My main goal for this project is to learn and get experience on processor working environment, implementation, and programming of a real project.

Tool chain

- Google SketchUp for the mechanical design of all the system, along with the use of Fusion 360 to create the required files for the 3D prints.
- My programming will be made in C++.
- Arduino IDE will be used to do the programming and the interaction between the PC and the ESP32 board.
- Kicad used to make the schematics for the project.
- FreeRTOS used to implement several tasks on the board at the same time.

José Ruben Garcia
BEE 526 Adv. Topics on Embedded Systems Design

Schedule

- Week 3 and 4 – Mechanical design and construction of base, finish research and order remaining parts and controller.
- Week 5 and 6 – Put all the parts together, code the behavior of base.
- Week 7 and 8 – Tests of movement and debugging.
- Week 9 and 10 – Add secondary features if possible.

Challenges

- Get familiar with processor interface and software.
- Design and access to 3D printer (Discovery Hall).
- Learn new technologies in a few weeks.
- Mechanical design of tracker and constraints.
- Keep the project on a low budget.

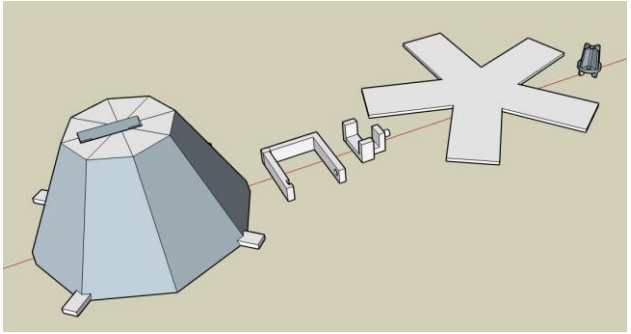
Progress over the weeks

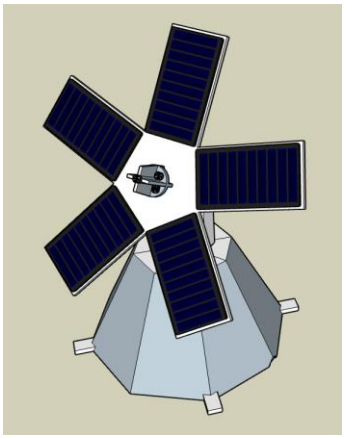
Week 3 (1st report)

- Have been researching similar projects and technologies.
- Changed the Raspberry pi Pico for a ESP32 in the need of more ADC inputs.
- Already ordered all the required elements for the project and I am waiting for materials to arrive.
- Got an Arduino UNO and a Raspberry pi 3 borrowed and have been testing them to get familiar with the Arduino IDE environment.
- Arduino IDE compatible with ESP32, so I can use the code when the ESP32 arrives.

Week 4 (2nd report)

- Received majority of the parts for the project.
- This week I did the mechanical design of the project.
- Will be 3D printing the required parts ASAP.
- Have been investigating and designing the secondary parts of the project: charging a battery bank with solar cells and using that energy to self-function; designing of a mobile-flower design.
- Started to code the behavior of the motors depending on the LDR light inputs.





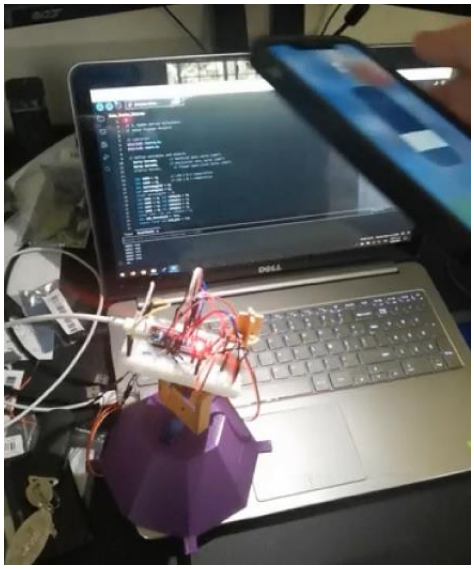
José Ruben Garcia
BEE 526 Adv. Topics on Embedded Systems Design

Week 5 (3rd report)

- Got 3D parts from Collaboratory.
- Finished coding the behavior of the solar tracker.
- Put together a “prototype design” of the system and the code is working properly moving the servo motors with the LDR signals.
- Investigating how to make both servo motor tasks to run simultaneously.

Week 6 (4th report)

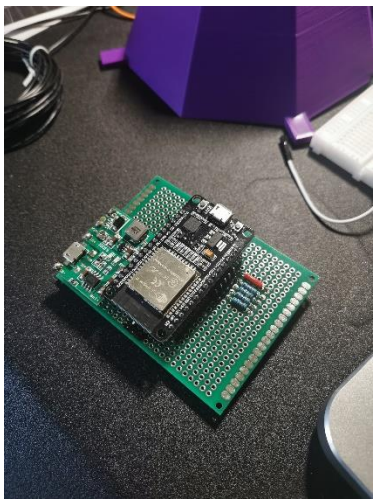
- Assembled part of the project, put together a prototype to test.
- Changed the processor board to an Arduino Nano, to get 5V output, as I had one laying around and the 3.3V of the ESP was giving me some trouble to move the servo motors.
- Had to change the code to fit my Arduino Nano.
- Started to do tests with the prototype, calibrated the amount of movement per step and the time of execution of the code.
- Found a bug when the horizontal axis servo passed 90° and the vertical servo would “flee” from the light at this point.

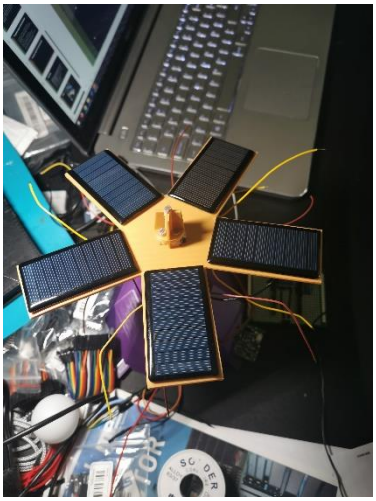


José Ruben Garcia
BEE 526 Adv. Topics on Embedded Systems Design

Week 7 (5th report)

- Tried to put a through hole PCB prototype, soldering skills not as good as I thought, need to do another one; also, I *supposed* how the level converter worked, and it didn't go as I was expecting, so I had to change the design.
- Thought I shorted and killed my ESP32 when I did some connections in my logic level converter wrong, did some tests and seems to be working fine.
- Been trying to implement the charger module with the solar array, my charger DCDC converter module is not working, ordered a new one.

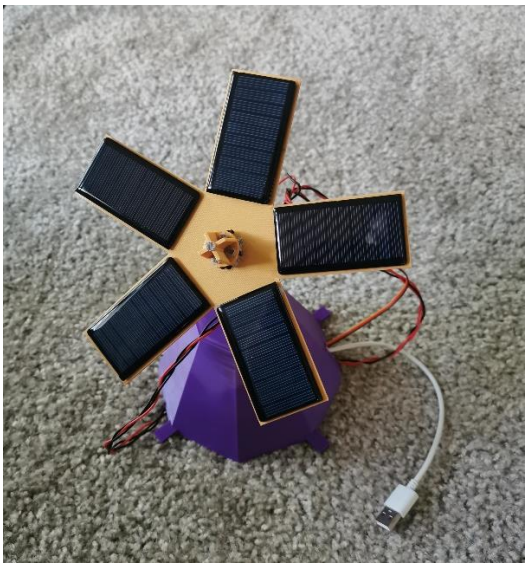


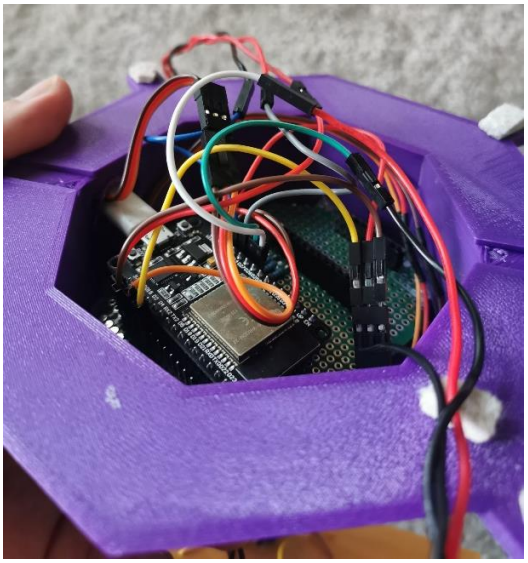


Week 8-9 (Final report)

- Faulty charger module, some short circuits in the board, servo not working properly.
- Did a new circuit board, everything working properly.
- Put everything together.
- Did testing with my RTOS program working perfectly, until I plugged my 2 servos to the board and realized the current delivered by the board was not going to be enough to power both servos at the time, had to change to the initial program moving one servo at a time.
- Discovered a faulty wire powering the LDRs, had to tear all the model apart and redo my wiring.
- After I did the wiring properly, the Horizontal axis servo stopped working.
- Switched it with a continuous rotating servo, as it was my only option at the moment, had to redo my code to fit my continuous servo as they work differently from the position servo.
- Got it working, not as smoothly as I would like for the horizontal axis but still.

José Ruben Garcia
BEE 526 Adv. Topics on Embedded Systems Design





Achievements

The main purpose of the project and characteristics that I achieved includes the following:

1. 2 axis light source follower (tracker).
2. Use of the ESP32 to control the movement of the base taking the LDRs inputs.
3. 3D design and 3D printed mechanical components.
4. Learnt to work on the Arduino IDE environment, as well as the use of the RTOS on the board using FreeRTOS on the board.
5. Learnt about processors, RTOS, I/Os, soldering, PCB design, schematics software, Arduino programming, PCB components for prototyping, different processor boards, servos, LCD screens (tried to make one work with the system at one time to measure output power), power design, different ways of coding depending on the components and the boards used, Libraries, DC motors, servo motors, soldering materials and components.

Unachieved

1. Secondary features, included: flower like design to mechanically open and close the solar array, charge/discharge module for a stand-alone system.
2. RTOS program running on the board to move both servos simultaneously. This caused because of the faulty charge/discharge module, leaving me without external supplies to power the servos. The current output coming from the board is not enough to move both servos at the time.
3. Smooth running servos, as my horizontal axis servo is not working to its full potential. I need to get another position servo to have a better control and a smoother movement. The vertical axis servo works without problems.
4. Couldn’t connect my solar array to the charge module, as it was faulty, to harvest energy.

Conclusions and thoughts

After this few weeks working in my project, a few thoughts come to mind when trying to do an analysis of the process that I went through to get to the point I am today.

If I could do the project again, I would take into account a few things:

- Try to focus on the main objective until its done; I spent a lot of time investigating and doing secondary features until I got not enough time to implement them.
- Ask for more help when encountering a problem, as it is really possible someone from the class will know the subject and will be willing to help.
- Start testing and debugging earlier.
- Try to extend the design stage, as a better design leads to less mistakes.

First of all, I'm happy with the result, as most of my goals were achieved and all the expected results were delivered properly.

Now, the project encountered its fair share of difficulties along the way, yet it found its way through effective problem-solving and determined work. Specially as it was a topic I wanted to dig in and explore using new tools and technologies that I wanted to understand.

The conducted weekly project progress reports and feedback sessions were a very important part for the success of this project, as it helped to broaden my vision of concepts I found difficult to master, enabling the refinement of the project.

José Ruben Garcia
BEE 526 Adv. Topics on Embedded Systems Design

Appendix

Code for Solar Tracker (First version, 1 servo motor moves at a time)

```
// ESP32 code

// Libraries
#include <Servo.h>
#include <math.h>

// Define variables and objects
  Servo ServoV;       // Vertical axis servo (180°)
  Servo ServoH;       // Horizontal axis servo (180°)
  //Servo ServoF;       // Flower open-close servo (360°)

  int LDRV = 0;       // LDR 1 & 2 comparative
  int LDRH = 0;       // LDR 3 & 4 comparative
  int servoAngleV = 0;
  int servoAngleH = 0;
  //int servoAngleF = 0;
  int LDR1 = 0; int LDR1pin = 13;
  int LDR2 = 0; int LDR2pin = 12;
  int LDR3 = 0; int LDR3pin = 14;
  int LDR4 = 0; int LDR4pin = 27;
  //int LDR5 = 0; int LDR5pin = 26;
  int LDR_threshold = 100;
  static const int LED_pin = LED_BUILTIN;
  int delaytasks = 15;

void setup() {
// Set serial port for communication
  Serial.begin(9600);
//  Serial.print("setup() running on core ");
//  Serial.println(xPortGetCoreID());

//Define I/O pins
  ServoV.attach(4);
  ServoH.attach(15);
  //ServoF.attach(4);

  pinMode (LDR1pin, INPUT);
  pinMode (LDR2pin, INPUT);
  pinMode (LDR3pin, INPUT);
  pinMode (LDR4pin, INPUT);
  //pinMode (LDR5pin, INPUT);
  pinMode (LED_pin, OUTPUT);
```

```
  servoAngleV = 0;
  servoAngleH = 90;
  //servoAngleF = 0;
  ServoV.write(servoAngleV);
  ServoH.write(servoAngleH);
  //ServoF.write(servoAngleF);
  delay (3000);
}

void loop() {
//// Read values from LDR5 to open/close flower
  //LDR5 = analogRead (LDR5pin);
  //if (LDR5 >= 400){analogWrite(servoAngleF = 288);}
  //else {analogWrite (servoAngleF = 0);}
  //delay(1000);

// Read values from position LDRs  0-4095
  LDR1 = analogRead (LDR1pin);
  LDR2 = analogRead (LDR2pin);
  LDR3 = analogRead (LDR3pin);
  LDR4 = analogRead (LDR4pin);

// LDR array comparative to define movement in H horizontal and V vertical
axis
  LDRH = abs(LDR1 - LDR3);
  LDRV = abs(LDR2 - LDR4);

// When one or both of the array comparatives is bigger than the threshold
start the servo's movement
// Builtin LED will be on when working
  if (LDRH > LDR_threshold || LDRV > LDR_threshold){
    digitalWrite(LED_pin, HIGH);
    if (LDRH > LDRV){
      if(LDR1 < LDR3 & servoAngleH < 160) // Not fully to 180° to avoid
contact between "flower" and base
        {servoAngleH = servoAngleH + 2;
          ServoH.write (servoAngleH);}
      if(LDR1 > LDR3 & servoAngleH > 20)  // Not fully to 0° to avoid
contact between "flower" and base
        {servoAngleH = servoAngleH - 2;
          ServoH.write (servoAngleH);}
  }
    if (LDRV > LDRH){
```

```
      digitalWrite(LED_pin, HIGH);
      if(LDR2 < LDR4 & servoAngleV < 180 & servoAngleH >= 90)  // Conditions
revert for V axis when surpassing the 90° threshold
        {servoAngleV = servoAngleV + 2;
          ServoV.write (servoAngleV);}
      if(LDR2 < LDR4 & servoAngleV > 0 & servoAngleH < 90)
        {servoAngleV = servoAngleV - 2;
          ServoV.write (servoAngleV);}
      if(LDR2 > LDR4 & servoAngleV > 0 & servoAngleH >= 90)
        {servoAngleV = servoAngleV - 2;
          ServoV.write (servoAngleV);}
      if(LDR2 > LDR4 & servoAngleV < 180 & servoAngleH < 90)
        {servoAngleV = servoAngleV + 2;
          ServoV.write (servoAngleV);}
  }
}
// Builtin LED turns off when not working
  if (LDRH < LDR_threshold & LDRV < LDR_threshold){digitalWrite(LED_pin,
LOW);}

//// Identify cores used to work
//  Serial.print("loop() running on core ");
//  Serial.println(xPortGetCoreID());
  Serial.print("LDR1 ");
  Serial.println(LDR1);
  Serial.print("LDR3 ");
  Serial.println(LDR3);
  Serial.print("LDR2 ");
  Serial.println(LDR2);
  Serial.print("LDR4 ");
  Serial.println(LDR4);

delay(delaytasks);
}
```

Code for Solar Tracker (Second version, RTOS running, both servos move at the same time)

```
// ESP32 RTOS code

// Libraries
#include <Servo.h>
#include <math.h>

// Tasks
```

```
TaskHandle_t Task1;
TaskHandle_t Task2;
TaskHandle_t Task3;

// Define variables and objects
  Servo ServoV;       // Vertical axis servo (180°)
  Servo ServoH;       // Horizontal axis servo (180°)
  //Servo ServoF;       // Flower open-close servo (360°)

  int LDRV = 0;       // LDR 2 & 4 comparative
  int LDRH = 0;       // LDR 1 & 3 comparative
  int servoAngleV = 0;
  int servoAngleH = 0;
  //int servoAngleF = 0;
  int LDR1 = 0; int LDR1pin = 13;
  int LDR2 = 0; int LDR2pin = 12;
  int LDR3 = 0; int LDR3pin = 14;
  int LDR4 = 0; int LDR4pin = 27;
  //int LDR5 = 0; int LDR5pin = 26;
  int LDR_threshold = 100;
  static const int LED_pin = LED_BUILTIN;
  int delaytasks = 15;



void setup() {
// Set serial port for communication
  Serial.begin(9600);
  Serial.print("setup() running on core ");
  Serial.println(xPortGetCoreID());

//Define I/O pins
  ServoV.attach(4);
  ServoH.attach(15);
  //ServoF.attach(4);

  pinMode (LDR1pin, INPUT);
  pinMode (LDR2pin, INPUT);
  pinMode (LDR3pin, INPUT);
  pinMode (LDR4pin, INPUT);
  //pinMode (LDR5pin, INPUT);
  pinMode (LED_pin, OUTPUT);

  servoAngleV = 0;
```

```
  servoAngleH = 90;
  //servoAngleF = 0;
  ServoV.write(servoAngleV);
  ServoH.write(servoAngleH);
  //ServoF.write(servoAngleF);
  delay (3000);

//Task 1 to run forever
  xTaskCreatePinnedToCore(
    ServoVertical,                // Function to be called
    "Servo Vertical Axis",        // Name of Task
    1024,                         // Stack size (bytes in ESP32, words in
FreeRTOS)
    NULL,                         // Parameter to pass the function
    1,                            // Task priority
    &Task1,                       // Task handle
    0);                           // Run on core 0
    delay(1000);

//Task 2 to run forever
  xTaskCreatePinnedToCore(
    ServoHorizontal,              // Function to be called
    "Servo Horizontal Axis",      // Name of Task
    1024,                         // Stack size (bytes in ESP32, words in
FreeRTOS)
    NULL,                         // Parameter to pass the function
    1,                            // Task priority
    &Task2,                       // Task handle
    1);                           // Run on core 1
    delay(1000);

//Task 3 to run forever
  xTaskCreatePinnedToCore(
    LEDoff,                       // Function to be called
    "Turn LED off",               // Name of Task
    1024,                         // Stack size (bytes in ESP32, words in
FreeRTOS)
    NULL,                         // Parameter to pass the function
    2,                            // Task priority
    &Task3,                       // Task handle
    1);                           // Run on core 1
    delay(1000);
}
```

José Ruben Garcia
BEE 526 Adv. Topics on Embedded Systems Design

```
// Task 1: Vertical axis servo
void ServoVertical(void *parameter){
  Serial.print("Task1 running on core ");
  Serial.println(xPortGetCoreID());

  while(1) {
// Read values from position 2 and 4 LDRs  0-4095
  LDR2 = analogRead (LDR2pin);
  LDR4 = analogRead (LDR4pin);

// LDR array comparative to define movement in V vertical axis
  LDRV = abs(LDR2 - LDR4);

// When the array comparatives is bigger than the threshold start the
servo's movement
// Builtin LED will be on when working
 if (LDRV > LDR_threshold){
    digitalWrite(LED_pin, HIGH);
      if(LDR2 < LDR4 & servoAngleV < 180 & servoAngleH >= 90)  // Conditions
revert for V axis when surpassing the 90° threshold
        {servoAngleV = servoAngleV + 2;
          ServoV.write (servoAngleV);}
      if(LDR2 < LDR4 & servoAngleV > 0 & servoAngleH < 90)
        {servoAngleV = servoAngleV - 2;
          ServoV.write (servoAngleV);}
      if(LDR2 > LDR4 & servoAngleV > 0 & servoAngleH >= 90)
        {servoAngleV = servoAngleV - 2;
          ServoV.write (servoAngleV);}
      if(LDR2 > LDR4 & servoAngleV < 180 & servoAngleH < 90)
        {servoAngleV = servoAngleV + 2;
          ServoV.write (servoAngleV);}
  }
  Serial.print("LDR2 ");
  Serial.println(LDR2);
  Serial.print("LDR4 ");
  Serial.println(LDR4);
  delay(delaytasks);
}
}

// Task 2: Horizontal axis servo
void ServoHorizontal(void *parameter){
  Serial.print("Task2 running on core ");
  Serial.println(xPortGetCoreID());
```

```
  while(1) {
// Read values from position LDRs  0-4095
  LDR1 = analogRead (LDR1pin);
  LDR3 = analogRead (LDR3pin);

// LDR array comparative to define movement in H horizontal and V vertical
axis
  LDRH = abs(LDR1 - LDR3);

// When the array comparatives is bigger than the threshold start the
servo's movement
// Builtin LED will be on when working
  if (LDRH > LDR_threshold){
    digitalWrite(LED_pin, HIGH);
      if(LDR1 < LDR3 & servoAngleH < 160) // Not fully to 180° to avoid
contact between "flower" and base
        {servoAngleH = servoAngleH + 2;
          ServoH.write (servoAngleH);}
      if(LDR1 > LDR3 & servoAngleH > 20)  // Not fully to 0° to avoid
contact between "flower" and base
        {servoAngleH = servoAngleH - 2;
          ServoH.write (servoAngleH);}
  }
  Serial.print("LDR1 ");
  Serial.println(LDR1);
  Serial.print("LDR3 ");
  Serial.println(LDR3);
  delay(delaytasks);
}
}

// Task 3: Turn off LED when not working
void LEDoff(void *parameter){
  Serial.print("Task3 running on core ");
  Serial.println(xPortGetCoreID());

  while(1) {
// Turn off the LED when servos not working
  if (LDRH <= LDR_threshold & LDRV <= LDR_threshold){digitalWrite(LED_pin,
LOW);}

  Serial.print("LED in Board ");
  Serial.println(digitalRead(LED_pin));
```

```
  delay(delaytasks);x
}
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Code for Solar Tracker (Third version, 1 servo motor moves at a time,1 cont. servo motor)

```
// ESP32 code

// Libraries
#include <Servo.h>
#include <math.h>

// Define variables and objects
  Servo ServoV;       // Vertical axis servo (180°)
  Servo ServoH;       // Horizontal axis servo (180°)
  //Servo ServoF;       // Flower open-close servo (360°)

  int LDRV = 0;       // LDR 1 & 2 comparative
  int LDRH = 0;       // LDR 3 & 4 comparative
  int servoAngleV = 0;
  int servoAngleH = 0;
  //int servoAngleF = 0;
  int LDR1 = 0; int LDR1pin = 13;
  int LDR2 = 0; int LDR2pin = 12;
  int LDR3 = 0; int LDR3pin = 14;
  int LDR4 = 0; int LDR4pin = 27;
  //int LDR5 = 0; int LDR5pin = 26;
  int LDR_threshold = 200;
  static const int LED_pin = LED_BUILTIN;

void setup() {
// Set serial port for communication
//  Serial.begin(9600);

//Define I/O pins
  ServoV.attach(4);
  ServoH.attach(15);
  //ServoF.attach(4);
```

```
  pinMode (LDR1pin, INPUT);
  pinMode (LDR2pin, INPUT);
  pinMode (LDR3pin, INPUT);
  pinMode (LDR4pin, INPUT);
  //pinMode (LDR5pin, INPUT);
  pinMode (LED_pin, OUTPUT);

  servoAngleV = 180;
  servoAngleH = 90;
  //servoAngleF = 0;
ServoV.write(servoAngleV);
ServoH.write(servoAngleH);
  //ServoF.write(servoAngleF);
  delay (3000);
}

void loop() {
//// Read values from LDR5 to open/close flower
  //LDR5 = analogRead (LDR5pin);
  //if (LDR5 >= 400){analogWrite(servoAngleF = 288);}
  //else {analogWrite (servoAngleF = 0);}
  //delay(1000);

// Read values from position LDRs  0-4095
  LDR1 = analogRead (LDR1pin);
  LDR2 = analogRead (LDR2pin);
  LDR3 = analogRead (LDR3pin);
  LDR4 = analogRead (LDR4pin);

// LDR array comparative to define movement in H horizontal and V vertical
axis
  LDRH = abs(LDR1 - LDR3);
  LDRV = abs(LDR2 - LDR4);
  if (LDRH < LDR_threshold) {ServoH.write (90);}
// When one or both of the array comparatives is bigger than the threshold
start the servo's movement
// Builtin LED will be on when working
  if (LDRH > LDR_threshold || LDRV > LDR_threshold){
    digitalWrite(LED_pin, HIGH);
    if (LDRH > LDRV){
      if(LDR1 < LDR3) // Rotate clockwise
        {ServoH.write (73);}
      if(LDR1 > LDR3)  // Rotate counter-clockwise
        {ServoH.write (118);}
```

```
    }
    else if (LDRH < LDRV){
      ServoH.write (90);
      digitalWrite(LED_pin, HIGH);
      if(LDR2 < LDR4 & servoAngleV < 180)  // Conditions revert for V axis
when surpassing the 90° threshold
        {servoAngleV = servoAngleV + 2;
          ServoV.write (servoAngleV);}
      if(LDR2 > LDR4 & servoAngleV > 0)
        {servoAngleV = servoAngleV - 2;
          ServoV.write (servoAngleV);}
    }
  }
// Builtin LED turns off when not working
  if (LDRH <= LDR_threshold & LDRV <= LDR_threshold) {digitalWrite(LED_pin,
LOW);}



//// Identify cores used to work
//  Serial.print("loop() running on core ");
//  Serial.println(xPortGetCoreID());
// Serial.print("LDR1 ");
// Serial.println(LDR1);
// Serial.print("LDR3 ");
// Serial.println(LDR3);
// Serial.print("LDR2 ");
// Serial.println(LDR2);
// Serial.print("LDR4 ");
// Serial.println(LDR4);

delay(20);
}
```