



Solar Link (Parte: Solar Tracker)

Carpeta Técnica

Adell Nicolas Fabian

Gil Soria Ian Lucas

García Rabal Lisandro

Suarez Tudisca Simon



Índice

1. Preámbulo	2
1.1. ¿Quiénes somos?	2
1.2. Docentes a cargo	2
1.3 Información adicional.....	3
1.3.1. Tiempo invertido	4
1.3.2. Programas utilizados	4
1.3.3. Lenguaje de programación utilizado	4
2. Introducción	5
2.1.Nombre de “Solar Tracker”	5
2.2. Resumen del proyecto.....	5
2.3. Utilización	5
3. Desarrollo técnico.....	6
3.1. Funcionamiento.....	6
3.2. Hardware	6
3.2.1. Descripción y periféricos utilizados	6
3.2.2 Composicion	7
3.2.3 Montaje	7
3.3. Software	8
3.3.1. Organización	8
3.3.2. Archivo “init.c”	8
3.3.3. Archivo “panel.c”	10
3.3.4. Archivo “main.c”	14
3.3.5. Archivo “CMake.Lists.txt”	16
3.4. Estructura	17
3.4.1.Concepto General.....	17
3.4.2. Conformación de la estructura.....	18

4. Listado de materiales	21
5. Referencias	23

1. Preámbulo

1.1. ¿Quiénes somos?

Nicolas Fabian Adell

DNI: 47020471

MAIL: nicolas.fabian2005@gmail.com

LINKEDIN: <https://ar.linkedin.com/in/nicolas-adell-354508297>

OFICIO: Software diseño y confección de hardware

Gil Soria Ian

DNI: 47337154

MAIL: ianlucasgilsoria@hotmail.com

LINKEDIN: www.linkedin.com/in/gil-soria-ian-070799226

OFICIO: Diseño y confección hardware y estructural



Suarez Tudisca Simón

DNI: 47294331

MAIL: simon.suareztudisca@gmail.com

OFICIO: Diseño estructural, soporte de hardware.



Lisandro García Rabal

DNI: 47350210

MAIL: lisandrogarciarabal@gmail.com

OFICIO: Confeccionamiento estructural y hardware general

1.2. Docentes a cargo

- **Carlassara Fabricio:**

Asistencia y guía en el desarrollo del Software

- **Daniel Espósito:**

Guía en el diseño del hardware y estructural

1.3. Información adicional

1.3.1. Tiempo invertido

- **Fecha de inicio:** 14 de abril de 2023
- **Duración:** 24 semanas de trabajo.
- **Esfuerzo del proyecto individual:** de 2 a 4 horas de trabajo semanales por integrante en relación con la habilitación del acceso al trabajo.
- **Esfuerzo total del proyecto:** 82 horas de trabajo.

1.3.2. Programas utilizados

- **Visual Studio Code:** Fue utilizado para la programación de la Raspberry Pi Pico, con la extensión del “CMake” para poder seguir las instrucciones para construir el programa
- **KiCad:** Fue utilizado para la confección de la placa del circuito impreso
- **Microsoft Office Word:** Fue utilizado para el desarrollo de la documentación
- **Google Chrome:** Fue utilizado como sitio para consultas e ideas sobre el proyecto, al igual que como fuente de aprendizaje.

1.3.3. Lenguaje de programación utilizado

- **C:** Fue el lenguaje de programación utilizado para la programación de la Raspberry Pi Pico

2. Introducción

2.1. Nombre de “Solar Tracker”

Decidimos ponerle este nombre a esta parte del proyecto debido a que la intención con esta parte del proyecto es que el panel intente seguir al sol para la obtención de su energía lumínica para convertirla en la energía eléctrica que la parte doméstica vaya a utilizar.

2.2. Resumen del proyecto

La metodología implementada conocida como “Solar Tracker” implica un traslado constante en el accionar del sistema sobre el panel ubicado sobre un eje rotativo mediante el cual su objetivo principal es evitar el desperdicio energético tanto absorbente por las células fotovoltaicas del mismo panel, con en la dirección y toma de valores energéticos máximos posibles, dando fe de un sistema automatizado, con el cual se busca maximizar la absorción energética solar (en su futuro, transformada en energía eléctrica)

Un ejemplo claro de la función que realiza un sistema “Solar Tracker” es un girasol, del cual se toma modelo y toma de guía en base al movimiento realizado, en un solo eje, fijado en posición estática, que a su vez realiza mediante su flor (en este caso hablamos del panel en sí, junto a su eje rotativo) el seguimiento del sol:

Las diferencias existentes entre el ejemplo dado y el mismo conocido “Solar Tracker”, es que el mismo dedica su función a la búsqueda de la posición del Sol mediante un escaneo, con el cual luego, volverá al lugar de toma de valor máximo de energía y se mantendrá por cierto periodo de tiempo hasta volver a realizar este proceso recientemente mencionado

2.3. Utilización

Si bien el movimiento del panel implica un consumo energético, está dedicado a que el indicado anteriormente sea de bajo coste y a su vez al hablar de un sistema de energías renovables, como bien es conocido el funcionamiento de un panel solar en sí, está diseñado a manera tal que sea un circuito de

retroalimentación con el cual se evita la problemática de consumo por el movimiento del “Solar Tracker”, gracias a la batería a la cual se administra energía; que también suministra, diferencia sobre el resto su uso por su rígida obligatoriedad de estar en constante dedicación a hallar un foco principal y más óptimo del cual obtener la máxima energía solar posible

3. Desarrollo técnico

3.1. Funcionamiento

El funcionamiento de la parte del proyecto desarrollada en las siguientes divisiones del índice, consiste en el seguimiento de un panel solar al Sol, buscando de manera constante el valor de tensión máximo admisible, aunque con un periodo de refrigeración para la calibración y obtención de los máximos valores posibles de tensión, Dando por hecho, la simulación de un “Girasol”, esto acompañado de la mano al conmutador de tensión de línea con el mismo panel, se conectan al hogar, dando una alternativa de energía renovable y su función como medio de ahorro económico.

3.2. Hardware

3.2.1. Descripción y periféricos utilizados

La parte electrónica de esta parte del proyecto se basa en la forma en que el microcontrolador recibe la tensión dependiendo el lugar en que se posicione y la salida del PWM que debe emitir para el movimiento del panel. De esta manera, la entrada de ADC, la cual mide la tensión que el panel entrega, se debió implementar un divisor resistivo sumado a un diodo Zener que se encuentra en paralelo a la resistencia de salida. Así pues, dependiendo la tensión que le es llegada al panel, el microcontrolador está programado tal que, con ayuda del servo, mediante la utilización de un ancho de pulso PWM específico para cada posición, se pueda encontrar la posición de mayor tensión y posteriormente aplicar el control proporcional.

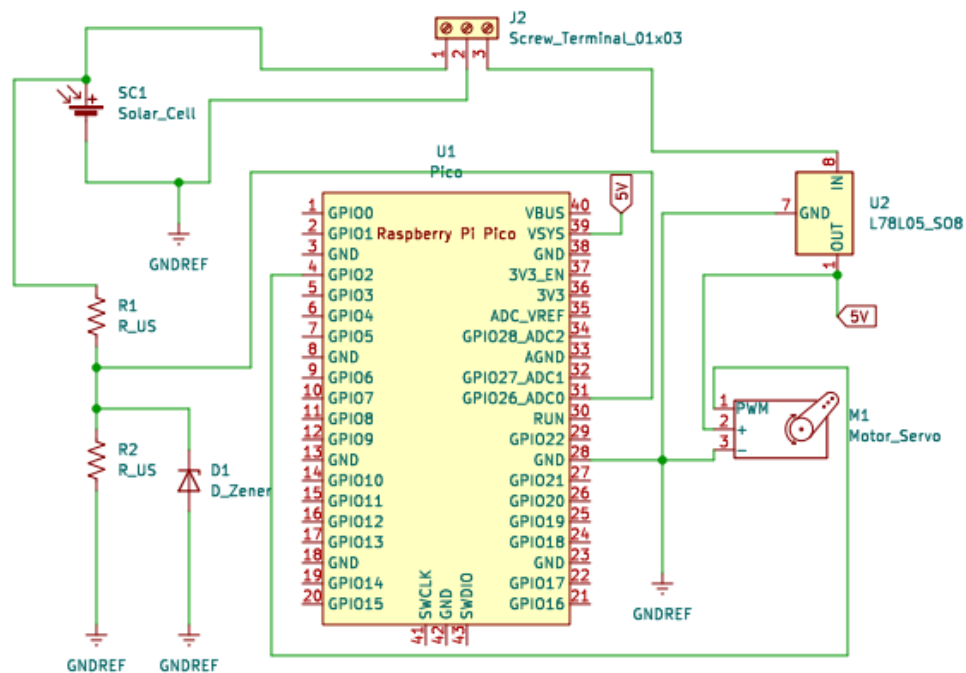
El pin de ADC utilizado es el GP26 o el ADC0, y el pin de PWM fue el GP2. Hay que tener en consideración que las masas

del circuito son todas iguales; es decir, que las masas del negativo de la batería, del panel y del circuito armado con el microcontrolador comparten la misma masa, permitiéndose esto debido al regulador MPPT realizado.

3.2.2. Composición

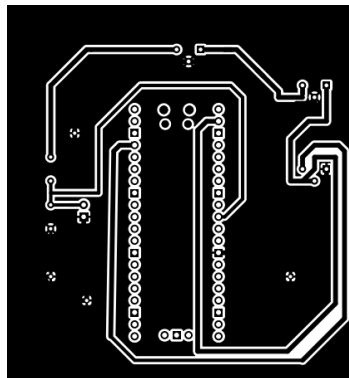
El circuito, por tanto, está compuesto por:

- Resistencias: una de 2,2 Kohm y otra de 10kOhm
- Diodo Zener: uno de tensión de umbral negativa de 3,3 V
- Panel: un panel KS20T
- Regulador: un L7805 (regulador de 5V)
- Raspberry Pi Pico: este es el microcontrolador programado para que siga la lógica correspondiente para seguir al sol
- Servo motor: un MG995 que es capaz de generar un torque de 10 kg/cm



3.2.3. Montaje

Luego de haber hecho el esquemático y haberlo probado en un protoboard se debió construir una placa de circuito impreso la cual fue diseñada en “KiCad”, para pasarla al circuito y luego pasarla por el percloruro férrico. Adjuntamos imagen del circuito que fue pasado a la placa de cobre:



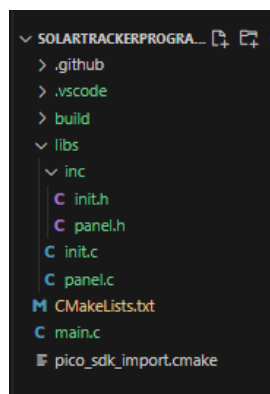
Circuito realizado en Kicad

3.3. Software

3.3.1. Organización

Con respecto al software, el lenguaje de programación utilizado fue C, utilizando el SDK para la Raspberry Pi Pico. Este mismo está dividido en tres archivos distintos, denominados “main.c”, “init.c” y “panel.c”.

El programa esta armado de manera tal de que las funciones llamadas en el “main” puedan acceder a los prototipos de funciones de los otros archivos además de la porción de código que representan.



Organización del template

3.3.2. Archivo “init.c”

En el init.c se desarrollaron las funciones pertenecientes a la inicialización del ADC y al PWM, ya que para este último se debe cargar una frecuencia específica de 50 Hz con el objeto de que, con las variaciones de ancho de pulso, el servo motor se pueda mover a la posición deseada. Esta frecuencia se puede obtener dividiendo la frecuencia del clock de la CPU con el prescaler correspondiente, además de establecer un wrap para la cantidad de posiciones. En este archivo, además de todo, se desarrollan las funciones correspondientes al movimiento del servo; es decir, se describen funciones tal que un ángulo represente una cantidad del ancho de pulso o viceversa. Así pues, se facilita el programa para el resto del código.

El código del “init.c” es el siguiente:

```
#include "init.h"

void initialize_adc(uint8_t adc_pin, uint8_t adc_input) {
    // Initialize ADC
    adc_init();
    adc_gpio_init(adc_pin);
    adc_select_input(adc_input);
}

void initialize_pwm(uint8_t pwm_pin) {
    // Initialize PWM
    gpio_set_function(pwm_pin, GPIO_FUNC_PWM);
    uint8_t slice = pwm_gpio_to_slice_num(pwm_pin);

    pwm_config config = pwm_get_default_config();

    // Divide clock in 50
    pwm_config_set_clkdiv(&config, 50.0);

    // Divide the output in 50000 possibilities
    pwm_config_set_wrap(&config, 50000);

    // pwm_config_set_output_polarity(&config, true, false);

    pwm_init(slice, &config, false);

    pwm_set_gpio_level(pwm_pin, get_position_with_degrees(0));

    pwm_set_enabled(slice, true);
}
```

```

int get_position_with_degrees(float degree) {
    // Calculate position with degrees
    int position = degree / 180 * 5000 + 1250;
    return position;
}

float get_degrees_with_position(int position) {
    // Calculate degrees with position
    float degree = ((float) position - 1250) * 180 / 5000;
    return degree;
}

```

3.3.3. Archivo “panel.c”

En el panel.c se describen principalmente las funciones relacionadas con el movimiento del panel, el cálculo de tensión en una cierta posición en específico, la búsqueda del punto de mayor tensión y por último se encuentra la función relacionada con la idea del control proporcional.

La lógica con respecto a el apunte del panel al lugar donde hay mas tensión se relaciona a la idea en el lugar donde se ubica la posición donde el sol le da más luz. De esta forma, la idea es primeramente buscar el punto de mayor tensión; es decir, aquella posición en la que el panel recibe más luz. Para realizar esto, el panel debe realizar en primera instancia un paneo general para encontrar el lugar de mayor tensión y evitar, sobre todo, que el panel no se quede estancado en una zona donde hay sombra. Esto se va a realizar en un lapsus de una hora, con el objeto de obtener el lugar de mayor tensión, aunque no realizarlo muy seguido para no desaprovechar la energía con la que el panel carga a la batería.

Posterior al paneo general, se encuentra la lógica del control proporcional. El control se basa en que, en primer lugar, el panel se mueve hacia un lado (horario si se encuentra en la posición más cercana a 90°, y antihorario si se encuentra en la posición más cercana a 0°). Luego de eso, se compara esa nueva posición con respecto a la anterior: si la tensión es superior, se elige esa posición como nuevo centro; en cambio, si la tensión medida en el ADC es menor, se vuelve a la posición original y se mueve el panel en el sentido contrario, así de alguna forma se queda comparando valores con el objeto de conseguir la mayor tensión posible sin

perder ninguna posición; además de esto, la medición de tensión se debe ir actualizando a medida que el panel se va moviendo. Es importante notar que la posiciones que el sol va tomando no son demasiado notables durante el día, o sea sus movimientos no son demasiado bruscos, salvo que aparezca una sombra, en cuyo caso se deberá esperar al siguiente paneo.

De esta forma, el panel podrá conseguir su objetivo que es seguir al sol de forma tal que obtiene la mejor tensión para ser aprovechada por la batería.

El código del “panel.c” es el siguiente:

```
#include "init.h"
#include "panel.h"

// Set variables to 0
float v_input_panel = 0;
float v_max_searching = 0;
bool after_searching_flag = false;
int current_position = 1250;
int panel_position_max_v = 1250;
uint8_t step_for_prop_control = 60;
bool clockwise_moved = false;
bool anticlockwise_moved = false;

float calculate_input_panel_voltage() {
    // Read tension of the ADC
    uint16_t adc_value = adc_read();

    // Calculate voltage from adc
    float voltage = adc_value * 3.3 / 4095;

    // Calculate input voltage from panel
    float v_input_panel = voltage * (10000 + 2200) / 2200;
    return v_input_panel;
}

void prop_control(uint8_t pwm_pin) {
    v_input_panel = 0;

    // Check if the panel has recently turned
    if (after_searching_flag == true) {
        first_movement_prop_control(pwm_pin);
        after_searching_flag = false;
    }
}
```

```

// Calculate voltage in the current position
v_input_panel = calculate_input_panel_voltage();
// Calculate the error
float error_calc = v_max_searching - v_input_panel;

printf("Tension max = %f\n", v_max_searching);
printf("Pos tension max = %i\n", panel_position_max_v);
printf("Pos actual = %d\n", current_position);
printf("Tension actual = %f\n", v_input_panel);
printf("-----\n");

// If error is greater, then return to the original position
if (error_calc >= 0) {
    printf("Tension menor al maximo\n");

    move_panel(pwm_pin, panel_position_max_v);

    v_max_searching = calculate_input_panel_voltage();

    printf("Nueva tension max = %f\n", v_max_searching);
    printf("Nueva pos tension max = %i\n", panel_position_max_v);
    printf("Nueva pos actual = %d\n", current_position);
    printf("-----\n");

    sleep_ms(30000000);
} else if (error_calc < 0) {

    printf("Tension mayor al maximo\n");

    // If error is lower, then a new maximum is set
    panel_position_max_v = current_position;
    v_max_searching = v_input_panel;
}

// It should alternate between clock and anticlockwise, despite
having an extreme angle

if (anticlockwise_moved == true ||
(get_degrees_with_position(current_position) >= (90 -
get_degrees_with_position(step_for_prop_control)))) {
    move_panel(pwm_pin, current_position - step_for_prop_control);
    anticlockwise_moved = false;
    clockwise_moved = true;
} else if (clockwise_moved == true ||
(get_degrees_with_position(current_position) <= (0 +
get_degrees_with_position(step_for_prop_control)))) {
    move_panel(pwm_pin, current_position + step_for_prop_control);

```

```

        anticlockwise_moved = true;
        clockwise_moved = false;
    }
}

void first_movement_prop_control(uint8_t pwm_pin) {
    // Save current position
    panel_position_max_v = current_position;
    v_max_searching = calculate_input_panel_voltage();

    printf("Primer tension: %f", v_max_searching);

    // Turn it one position to start the proportional control
    if (get_degrees_with_position(current_position) >= 0 &&
get_degrees_with_position(current_position) <= 45) {
        move_panel(pwm_pin, current_position + step_for_prop_control);
        anticlockwise_moved = true;
        clockwise_moved = false;
    } else if (get_degrees_with_position(current_position) > 45 &&
get_degrees_with_position(current_position) <= 90) {
        move_panel(pwm_pin, current_position - step_for_prop_control);
        clockwise_moved = true;
        anticlockwise_moved = false;
    }
}

void move_panel(uint8_t pwm_pin, int desired_position) {

    // Check if the new position is possible
    if (get_degrees_with_position(desired_position) < 0 ||
get_degrees_with_position(desired_position) > 90
        || desired_position == current_position) {
        return;
    }

    // Move one position at a time to reach the disired position
    while (current_position != desired_position) {

        int new_position;
        if (desired_position - current_position > 0) {
            new_position = current_position + 1;
            pwm_set_gpio_level(pwm_pin, new_position);
        } else if (desired_position - current_position < 0) {
            new_position = current_position - 1;
            pwm_set_gpio_level(pwm_pin, new_position);
        }

        current_position = new_position;
        sleep_ms(5);
    }
}

```

```

    }
}

void search_of_max_voltage(uint8_t pwm_pin) {

    // Set the variables in 0 degrees
    int pos_variable = get_position_with_degrees(0);
    v_max_searching = 0;
    panel_position_max_v = get_position_with_degrees(0);

    // The loop continues up to 180 degrees
    do {
        // Move the panel to the new position
        move_panel(pwm_pin, pos_variable);
        // Calculate the input voltage
        v_input_panel = calculate_input_panel_voltage();
        // If it's grater than the last measure, the variable will be
saved
        if (v_input_panel > v_max_searching) {
            v_max_searching = v_input_panel;
            panel_position_max_v = current_position;
        }

        // Next position is set
        pos_variable++;
    } while (((get_degrees_with_position(pos_variable)) < 90));

    printf("Despues de buscar el maximo\n");
    printf("Posicion actual: %d\n", current_position);
    printf("Tension max = %f\n", v_max_searching);
    printf("Pos tension max = %i\n", panel_position_max_v);
    printf("Tension actual = %f\n", v_input_panel);
    printf("-----\n");

    // Set the flag when it's finished
    after_searching_flag = true;
}

```

3.3.4. Archivo “main.c”

Esta es la parte del código que se encarga de la orden de ejecución de las funciones, haciendo los pasos necesarios para que haga el paneo cada una hora y el posterior control proporcional.

El código del “main.c” es el siguiente:

```

#include <stdio.h>
#include "panel.h"
#include "init.h"
#include "pico/time.h"

int main(void) {
    // Set pins
    const uint8_t ADC_PIN = 26;
    const uint8_t ADC_INPUT = 0;
    const uint8_t PWM_PIN = 2;

    stdio_init_all();

    // Initialize ADC
    initialize_adc(ADC_PIN, ADC_INPUT);

    // Initialize PWM
    initialize_pwm(PWM_PIN);

    bool need_of_searching = true;
    absolute_time_t t0 = get_absolute_time();

    while (true) {

        // If the degree isn't inside the angles possible, then the panel
        // should go to 90 degrees
        if (get_degrees_with_position(current_position) < 0 ||
            get_degrees_with_position(current_position) > 90) {
            move_panel(PWM_PIN, get_position_with_degrees(45));
            need_of_searching = true;
        }

        // Do the searching every 30 minutes

        absolute_time_t t1 = get_absolute_time();

        // Set the time in which the searching should be executed
        if (absolute_time_diff_us(t0, t1) >= 3600000000) {
            need_of_searching = true;
        }

        if (need_of_searching == true) {
            search_of_max_voltage(PWM_PIN);
            t0 = get_absolute_time();
            // Move the panel to the position with the maximum voltage

```



```

        move_panel(PWM_PIN, panel_position_max_v);

        need_of_searching = false;
    }

    // After searching the maximum the panel will make the
    proportional control
    prop_control(PWM_PIN);

    sleep_ms(750);
}

return 0;
}

```

3.3.5. Archivo “CMakeLists.txt”

Este archivo es de gran importancia debido a que le da las instrucciones a la Raspberry Pi Pico de lo que tiene que llevar a cabo para construir el archivo en formato “.uf2”

El CMake está conformado por lo siguiente:

```

cmake_minimum_required(VERSION 3.13)
include(pico_sdk_import.cmake)
project(project_name)
pico_sdk_init()

add_library(libs
    libs/init.c
    libs/panel.c
)

target_include_directories(libs PUBLIC
    libs/inc
)

```

```
target_link_libraries(libs
    pico_stdlib
    hardware_adc
    hardware_pwm
)
```

```
add_executable(solarTrackerProgramming
    main.c
)
```

```
target_link_libraries(solarTrackerProgramming pico_stdlib pico_time
    hardware_adc hardware_pwm libs)
```

```
pico_enable_stdio_usb(solarTrackerProgramming 1)
pico_enable_stdio_uart(solarTrackerProgramming 0)
```

```
pico_add_extra_outputs(solarTrackerProgramming)
```

3.4. Estructura

3.4.1. Concepto General

Esta consiste en una base conformada principalmente por la unión de 4 tubos de PVC en los cuales a sus esquinas se localizan fijaciones o bien conectores en forma de “Codos” (uniones a 90°), Dando una estructura si bien liviana pero resistente, a la vez, modificable su peso: con la habilidad de ser huecos, 17pudiéndose rellenarse de materiales entre los cuales posibles están Arcilla, Cemento (En base a circunstancias excepcionales externas). De la cual se adhiere una “H” donde será implementado y colocado en su centro el panel solar en sí.

Los requisitos dados como objetivos del armado destacan los siguientes puntos:

- Tener un medio por el cual pueda trabajar el programa integrado en el microcontrolador Raspberry Pi Pico.
- Dar por el hecho libertad de movimiento al eje del servomotor.
- Funcionar como sostén principal del diseño esquemático previamente realizado transformado a lo físico, de esta parte del proyecto tanto para su movimiento como superficie y esqueleto de la misma.

3.4.2. Conformación de la estructura

En la cara dirigida a la parte superior (si bien se puede hablar como cara superior a lo referido como la zona del tubo con la que se formase un ángulo de 90 grados en respecto a un eje x) de los tubos paralelos de 62 cm, a sus mitades, se hayan unos huecos para encastre dentro de los cuales se posicionan los tablones de madera, fijados y sellados con macilla química

Quedando así los tablones de madera en posición vertical a 90°, incrustados en los tubos de PVC dando forma de arco y ubicados paralelos entre si, quedan posicionados frente a frente las perforaciones realizadas en ambos tablones en sus mitades que permiten el paso de la varilla/cilindro de duraluminio



Se presenta la estructura en una etapa temprana y su forma básica

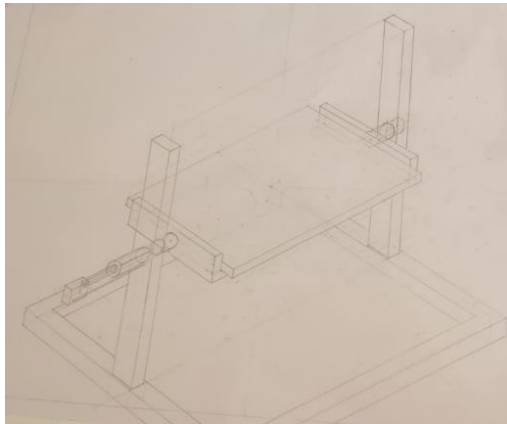
En el paso realizado por las perforaciones en donde atraviesa el cilindro de duraluminio se le colocan para que sufra de libre

movimiento la varilla rulemanes dentro de los tablones de madera incrustados, al extremo del cilindro (anotación: el mismo sobresale por ambos extremos de los tablones, dando seguridad de que no por un movimiento abrupto se desencastre el cilindro del arco formado por los tablones). Esta colocado el servomotor ajustado con macilla química para que el rotor del mismo le de el movimiento a la varilla; el servomotor esta ajustado al tablón del lado izquierda mediante dos tornillos autoperforantes para madera, a manera tal de que al realizar torque el servomotor, el mismo es contenido y fijado por estos.

El panel posee dos placas de duraluminio en sus extremos del ancho, que funcionan como vigas, ajustadas por pequeñas chapas en formas de L mediante perforaciones realizadas en el panel y en las chapas (como en las mismas uniones de forma “L”). A su vez, se les coloco en el centro de cada una, una regleta tipo abrazadera con huecos para tornillos, mediante los cuales se ajusta y traba el cilindro de duraluminio para ser fijado y sostenido el panel a la varilla; dando así la posibilidad de realizar movimientos con el servomotor dirigidos al panel mediante un método funcional



Se presenta la estructura completa junto al panel y el servomotor conectado



Se presenta un bosquejo tipo lamina esquemática a escala con la cual se determinaron las dimensiones de las principales partes de la estructura, así como su localización y función

Se presentan las medidas tomadas e ideadas, con sus respectivas conversiones realizadas a criterio propio en un sentido de escala 1/40 a escala real:

Cilindros:

Grande N°1 (Independiente):

Largo: $3 \text{ cm} = 3 * 40 = 120 \text{ mm} = 12 \text{ cm}$

Diámetro: $1 = 1 * 40 = 40 \text{ mm} = 4 \text{ cm}$

Grande N°2 (Conectado al chico):

Largo: $4 \text{ cm} = 4 * 40 = 160 \text{ mm} = 16 \text{ cm}$

Diámetro: $1 = 1 * 40 = 40 \text{ mm} = 4 \text{ cm}$

Chico:

Largo: $2 \text{ cm} = 2 * 40 = 80 \text{ mm} = 8 \text{ cm}$

Diámetro: $0,5 \text{ cm} = 0,5 * 40 = 20 \text{ mm} = 2 \text{ cm}$

Solapas en "U":

Bordes:

Largo: $7 \text{ cm} = 7 * 40 = 280 \text{ mm} = 28 \text{ cm}$

Alto: $1,5 \text{ cm} (\text{con saliente de } 0,5 \text{ cm}) = 1,5 * 40 = 60 \text{ mm} = 6 \text{ cm}$

Espesor: $0,5 \text{ cm} = 0,5 * 40 = 20 \text{ mm} = 2 \text{ cm}$

Plataforma:

Largo: $13 \text{ cm} = 13 * 40 = 520 \text{ mm} = 52 \text{ cm}$

Ancho: $7 \text{ cm} = 7 * 40 = 280 \text{ mm} = 28 \text{ cm}$

Espesor: $0,5 \text{ cm} = 0,5 * 40 = 20 \text{ mm} = 2 \text{ cm}$

Palos en "H":

Espesor: $0,875 \text{ cm} = 0,875 * 40 = 35 \text{ mm} = 3,5 \text{ cm}$

Largo: $15 \text{ cm} = 15 * 40 = 600 \text{ mm} = 60 \text{ cm}$

Ancho: $1,5 \text{ cm} = 1,5 * 40 = 60 \text{ mm} = 6 \text{ cm}$

Largo: $52 \text{ cm} = 520 \text{ mm}$

Ancho: $35,2 \text{ cm} = 352 \text{ mm}$

Base, escala 1/40:

Largo $18 \text{ cm} = 40 * 18 = 720 \text{ mm} = 72 \text{ cm}$

Diferencia de encastre (esquina) entre sí: 1 cm

$1 * 40 = 40 \text{ mm} = 4 \text{ cm}$

Ancho: $16 \text{ cm} = 40 * 16 = 640 \text{ mm} = 64 \text{ cm}$

Largo del caño: $242,5 \text{ cm}$

$72 \text{ cm} + 72 \text{ cm} + 64 \text{ cm} = 242,5 \text{ cm}$

$72 \text{ cm} + 72 \text{ cm} + 64 \text{ cm} + 64 \text{ cm} = 272 \text{ cm}$

(Restando diferencia de esquina = encastra justo sumando uniones en L)

$272 \text{ cm} - 16 \text{ cm} = 256 \text{ cm}$

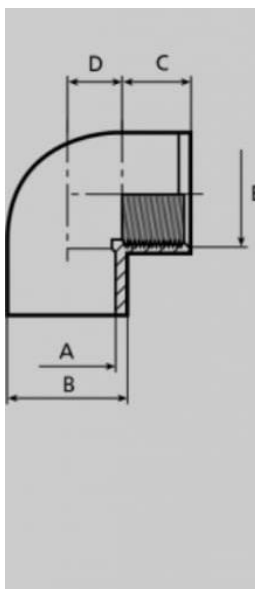
El proceso que se tomo fue el siguiente: La medida dentro de la lámina es realizada en centímetros, luego es multiplicada por el numero de la escala, el numero dado se lo toma como milímetros y se lo pasa centímetros

Aclaración: Las medidas a lo largo del plazo de realización de esta parte del proyecto sufrieron cambios

4. Listado de materiales

- Cilindro solido de material Duraluminio

- Dimensiones: 1 metro de Largo x 14,29 centímetros de diámetro, con perforaciones para atornillado de 2,5 mm
- 2 Rulemanes modelo: Rodamiento 6202 Rs. Power
- Dimensiones: 15 milímetros de diámetro interior x 35 milímetros de diámetro exterior x 11 milímetros de altura
- 2 Tornillos autoperforantes para madera
- Dimensiones: 2 ½ pulgadas
- 2 Placas de duraluminio:
- Dimensiones: 3 mm de espesor x 60 mm de ancho x 352 mm de largo
- 4 Uniones en “L” de duraluminio y hierro:
- Dimensiones: Las mismas dan forma de dos cuadrados con una apertura a 90°
- Uniones de material “PVC” del tipo “Codo” /” Curva”
- Dimensiones: Angulo de inclinación 90° x Diámetro de 40mm x “A” de 50 mm x “B” de 63mm x “C” de 31mm, “D” de 27mm y “E” de 1 ½ “– con un peso aproximado de 0,180 Kilogramos (Se adjunta Referencia Visual)
- Poxilina: 150 gramos
- 14 tornillos de 1 cm tipo cabeza de domo



DN	A	B	C	D	E	Kg.
10	16	22	14	9	3/8"	0,010
15	20	25	16	10	1/2"	0,020
20	25	32	19	14,75	3/4"	0,029
25	32	40	22	18,5	1"	0,050
32	40	50	26	23	1 1/4"	0,084
40	50	63	31	27	1 1/2"	0,180
50	63	75	38	32,5	2"	0,255
50	63	75	38	32,5	1 1/2"	0,274
65	75	90	44	40	2 1/2"	0,422
80	90	110	51	48	3"	0,786
100	110	131	61	58	4"	1,171

- 4 Tubos de material "PVC":

2 Tubos horizontales:

- Dimensiones: Diámetro de 40 mm x Largo de 70 cm

2 Tubos verticales:

- Dimensiones: Diámetro de 40 mm x Largo de 62 cm

- Dos Tablones de Madera tipo contrachapada con perforación:

- Dimensiones: Espesor de 1,75 cm x Largo de 60 cm (más 20 centímetros agregados) x Ancho de 6 cm

-Panel Solar Modelo "KS20T"

- Dimensiones: Largo de 520 mm x Ancho de 352 mm x Alto de 22 mm x Peso de 2,40 kilogramos
- Parámetros: Potencia Nominal (NP): 20 Wp x Voltaje NP 17,4 v x Corriente NP 1,16 A x Tensión de circuito abierto 21,7 v x Intensidad de Cortocircuito 1,26 A

- Resistencias: Una de 2,2 Kohm y otra de 10kOhm

- Diodo Zener: Uno de tensión de umbral negativa de 3,3 V

- Regulador: Un L7805 (regulador de 5V)

- Raspberry Pi Pico: Este es el microcontrolador programado para que siga la lógica correspondiente para seguir al sol

- Servo motor: un MG995 que es capaz de generar un torque de 10 kg/cm

5. Referencias

- <https://github.com/ncarandini/KiCad-RP-Pico/blob/main/Install%20instructions.md>
- <https://github.com/WexterHome/SolarTraker>

- https://www.researchgate.net/figure/prototype-of-a-single-axis-solar-tracking-system_fig3_348161146
- http://ve.scielo.org/scielo.php?script=sci_arttext&pid=S1316-48212006000300009
- <https://www.inventable.eu/2012/05/31/energia-solar-parte-3-el-sensor-de-tension-y-corriente/>
- <https://randomnerdtutorials.com/micropython-interrupts-esp32-esp8266/>