



ICAT
Instituto de Ciencias
Aplicadas y Tecnología

Reporte Final de Actividades Servicio Social



APOYO EN LOS PROYECTOS QUE SE DESARROLLAN EN EL ICAT

Elaboración de un sistema de control de seguimiento satelital para los rotores Yaesu G5400 – B



- Estudiante:** Iñaky Ordiales Caballero
- Tutor:** M.I. Rafael Prieto Meléndez
- Institución:** Instituto de Ciencias Aplicadas y Tecnología, UNAM
- Laboratorio:** Modelado y Simulación de Procesos
- Fecha inicio:** 27 de junio del 2022.
- Fecha final:** 20 de enero del 2023.

Índice:

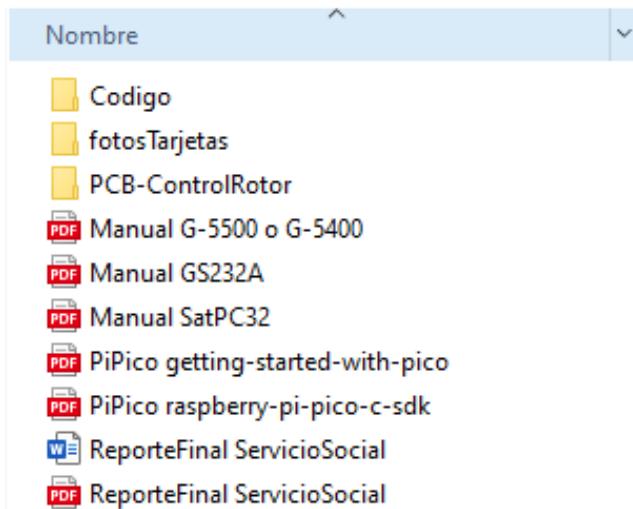
1. Introducción	02
2. Marco Teórico	03
3. Arquitectura y codificación Raspberry Pi Pico	08
4. Configuración de RP Pico	11
5. Compilar y cargar el programa	19
6. Prototipo	21
7. Diagramas	25
8. Materiales y armado	27
9. Configuraciones de Software	32
10. Modo de uso y conclusiones	40

1. Introducción

En este documento se presenta de manera resumida los resultados del trabajo realizado durante el servicio social que realicé en el Instituto de Ciencias Aplicadas y Tecnología de la UNAM durante los meses de junio del 2022 a enero del 2023. El objetivo de este reporte no es sólo dejar constancia de lo realizado, sino el permitir que otras personas que lo lean entiendan el proyecto elaborado, lo puedan replicar y de ser necesario modificar. Por lo que se explica desde el cómo configurar el entorno para poder desarrollar código para la Raspberry Pi Pico, el cómo compilarlo en lo general y nuestro proyecto en específico, hasta las configuraciones del software utilizado para obtener las posiciones de los satélites.

Para empezar, explicaré un poco sobre el proyecto. El propósito del proyecto era poder crear una interfaz a través de la cual se conectara un software que indicara el posicionamiento de las antenas para poder leer las transmisiones de algún satélite en específico. El sistema desarrollado debía poder recibir el posicionamiento mediante los datos de azimut y elevación para poder enviar las señales correspondientes a los rotores y posicionar las antenas en el lugar correcto. El modelo de rotor utilizado fue el YAESU G5400-B y el software por el que se optó fue el programa SatPC32 para Windows. De esta forma el proyecto consistió en desarrollar desde los prototipos en protoboard e impresión manual del PCB, hasta el armado final del sistema y sus periféricos en una caja. Claro que además se tuvo que desarrollar el código que la Pi Pico utiliza para entender la comunicación, determinar la posición actual y enviar las señales para poner las antenas en la posición deseada. A lo largo del documento se explica más a detalle cada una de las etapas y características.

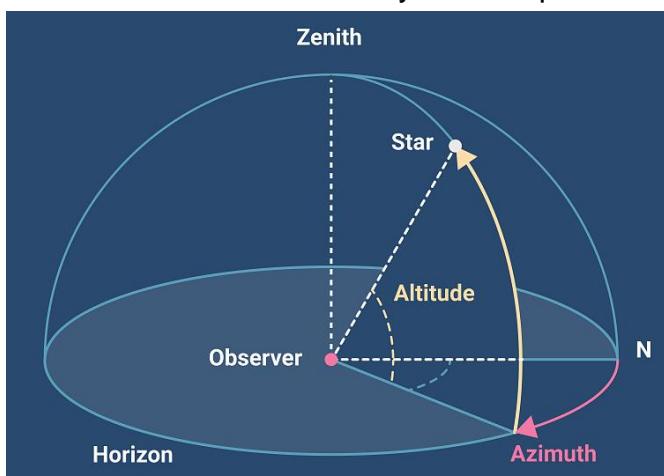
El entregable final del trabajo realizado en el servicio social consta de una carpeta que contiene los siguientes archivos:



- En el directorio **Codigo** están los códigos fuentes del programa desarrollado, así como las bibliotecas necesarias para su compilación y los binarios resultantes de la compilación.
- En el directorio **fotosTarjetas** se guardan imágenes de la tarjeta desde el primer prototipo hasta la versión final armada (mismas incluidas a lo largos de este documento).
- En el directorio **PCB-ControlRotor** se encuentran los archivos de KiCad 6.0 donde se realizó el diseño esquemático y de la PCB final del controlador de los motores.
- El archivo **Manual G-5500 o G-5400** es el manual del motor Yaesu utilizado.
- El archivo **Manual GS232A** es el manual del controlador Yaesu que se replicó.
- El archivo **PiPico getting-started-with-pico** es el instructivo de inicio para el desarrollo en C/C++ de la Raspberry Pi Pico.
- El archivo **PiPico raspberry-pi-pico-c-sdk** es una compilación de las bibliotecas disponibles en C/C++ para utilizar en el desarrollo.
- Y las versiones del **ReporteFinal ServicioSocial** es el mismo documento en versión final y versión editable que se está leyendo.

2. Marco Teórico

Para empezar con el desarrollo del proyecto tuve que aprender un poco sobre la terminología y características satelitales. Si bien mucha de la investigación inicial sólo sirvió como contexto y no era plenamente necesaria para llevar a cabo el



sistema requerido, sí me ayudó a comprender el propósito y la necesidad de lo que se iba a hacer. Realmente de la investigación inicial, lo más relevante fue el comprender lo que es el azimut y la elevación para poder direccionar adecuadamente las antenas. Fuera de esto aprendí sobre las diferentes bandas de frecuencias que se usan para transmitir información tanto a nivel aficionado

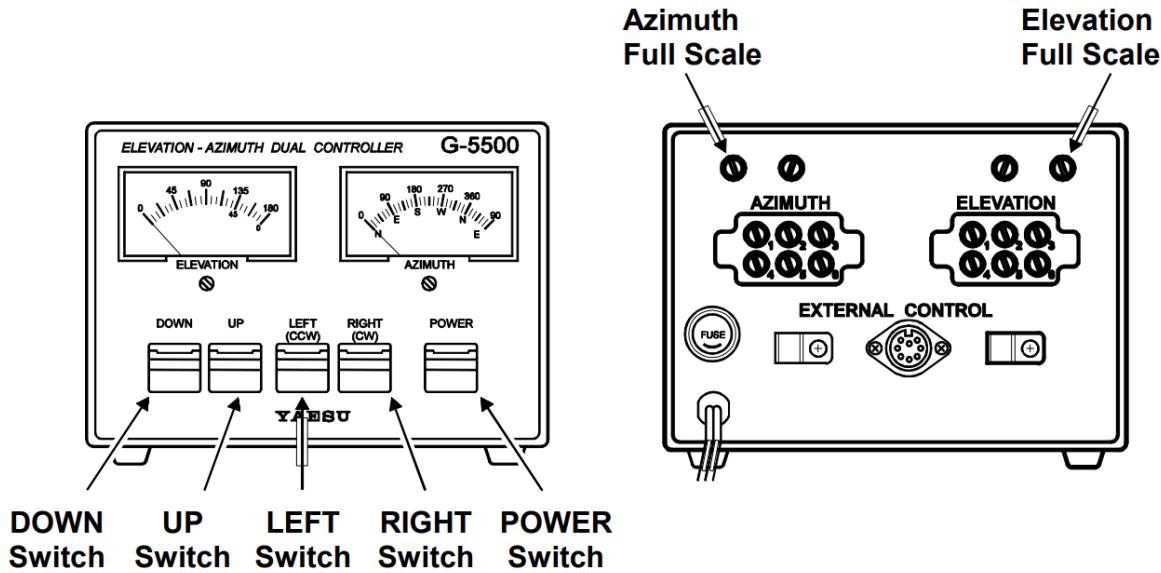
como ya en forma institucional. Leí un poco sobre el cálculo de las órbitas de los satélites y la razón por la cual se tienen que actualizar estos cálculos cada determinado tiempo. Por esto se optó por usar un software y no crearlo desde cero.

Rotor YAESU G-5400B:

Como parte de la información necesaria para la creación del sistema de control, se tuvo que leer el instructivo del rotor que se iba a utilizar. En este caso se usó el rotor de la marca Yaesu modelo G-5400B que comparte características con el modelo G-5500. Dentro de estas las más relevantes eran que el control externo se llevaba a cabo por medio de 8 cables:

- Voltaje directo de 13V.
- Tierra del circuito del rotor (GND).
- Voltaje correspondiente al azimut (0° a 360°)
- Voltaje correspondiente a la elevación (0° a 180°)
- Señal para mover hacia abajo (menor elevación)
- Señal para mover hacia arriba (mayor elevación)
- Señal para mover hacia la derecha (desde -180° hacia 180°)
- Señal para mover hacia la izquierda (desde 180° hacia -180°)

Además, se aprendió cómo calibrar el dispositivo tanto en la aguja analógica, como en la señal de voltaje para el posicionamiento. Como nuestro convertidor analógico a digital mide como máximo 3.3V se ajustó para este valor. Para mover los rotores con un control externo había que bajar a tierra las líneas de arriba, abajo, izquierda o derecha. Tarda 58 segundos en recorrer los 180° de elevación y 53 segundos en recorrer los 360° de azimut. A continuación, se muestra una imagen del instructivo donde se enseña cómo es el controlador de azimut y elevación de los rotores que mueven las antenas.



Control externo Yaesu GS-232A:

La misma marca que fabrica los rotores, también vende por separado el controlador externo para poder conectar los rotores a un programa en una computadora que envíe los comandos pertinentes a este dispositivo. Es importante mencionar que la mayoría de los softwares para rastrear satélites ya incluye la interfaz de comunicación con este dispositivo a través de un puerto serial. Debido a esto la idea del proyecto fue el replicar el funcionamiento de este controlador externo, para poder seleccionar su interfaz de comunicación en los programas de computadora y que la comunicación fuera exitosa con nuestra tarjeta que usa la Raspberry Pi Pico.



Para poder entender los comandos que se debían poder procesar y administrar en nuestra tarjeta, se revisó el manual del dispositivo Yaesu GS-232A y el código fuente de un programa libre llamado GPredict. Gracias a estos elementos se pudo entender el formato en que se recibían los comandos y su significado para poder replicarlo en el código de la Pi Pico. A continuación, se anexan los comandos que vienen en las hojas del manual del GS232A.

OPERATION	COMMAND LIST
	In the following command descriptions, the elevation version of each command, where there is one, is shown in parentheses (but don't type the parentheses). Remember that elevation commands require the G-5400B , G-5600B or G-5500 Az/El Rotators, or the GX-500 adapter and the G-500 or G-550 Elevation Rotator.
O (02)	R (U) Start turning the rotator to the right (up)
<i>Offset calibration</i> for internal AZ (EL) trimmer potentiometer: preset rotator manually fully counter-clockwise, send command, and adjust trimmer on Control Interface until returned values are equal. Turn off the GS-232A's POWER switch to store settings.	L (D) Start turning the rotator to the left (down).
H (H2)	A (E) Stop azimuth (elevation) rotation.
Returns list of commands (see page 19).	S Stop: cancel current command before completion.
F (F2)	C (B) Return current azimuth (elevation) angle in the form "+0nnn" degrees.
<i>Full Scale Calibration</i> : preset rotator manually to full scale, send command, adjust OUT VOL ADJ trimmer on rear of controller (or GX-500 elevation adapter) until the returned data is "+0180 or +0450" ("+0nnn+0180" for elevation). Turn off the GS-232A's POWER switch to save new settings.	C2 Return azimuth and elevation ("+0aaa+0eee", where <i>aaa</i> = azimuth, <i>eee</i> = elevation).
	Xn Select azimuth rotator turning speed, where <i>n</i> = 1 (slowest) to 4 (fastest). This command can be issued during rotation, and takes effect immediately. There is no equivalent for elevation.

COMMAND LIST

Maaa

Turn to **aaa** degrees azimuth, where aaa is three digits between “000” and “360 or 450: vary according to controller type.” Rotation starts.

Msss aaa bbb ccc

This command, together with the [T] command, provides automatic, timed tracking of moving objects or propagation by the Control Interface itself. This command stores the time value **sss** seconds to wait between stepping from azimuth **aaa** to **bbb**, and then to **ccc**, etc. (from “2” to as many as “3800” angles may be stored with one command).

Note that this command is completely different than the [T] command with only one parameter: when multiple parameters are present, the first one is interpreted by the Control Interface as the rotation interval **sss**, not an angle. Valid ranges are “001” to “999” for **sss**, and “000” to “360 or 450: vary according to controller type” for the angles. When this command is sent, the parameters are stored in the Control Interface’s RAM, and the rotator turns to angle **aaa** and waits for a subsequent [T] command to begin the actual stepping. All numbers must be 3 digits, space-separated. Stored values remain in effect until another [M] command is issued (this may have no parameters, in which case the “? >” error prompt is returned, but memories are still cleared), or until the controller is turned off or by toggling the **GS-232A** off and on.

T

See the [M] (above) and the [W] (below) command. Start automatic stepping routine (both azimuth and elevation): turn rotator to next sequentially memorized azimuth (or az-el pair, for the [W] command), wait **sss** seconds, and turn to next angle (or pair), etc. This command works only if a long-form [M] or [W] has been issued since power-up or the last reset.

N

Return serial number of currently selected memorized point [**nnnn**], and total number of memorized points [**mmmm**], in the form +**nnnn+mmmm**. Must be proceeded by either a long-form [M] or [W], and a T command. Used only during stepping (see [T] command).

The meaning of a “point” in this command following an [M] command is only an azimuth angle, so in this case **nnnn** and **mmmm** can range up to “3800” (the limit of available RAM in the Control Interface). However, when elevation is involved, a “point” following a [W] command is represented by both an azimuth and an elevation angle, in which case **nnnn** and **mmmm** can range up to only “1900,” since each “point” is a pair of angles.

17

COMMAND LIST

Elevation Control Commands

These commands are only for az-el operation. Note that an azimuth angle must always be supplied when changing elevation, and that a setting point consists of a pair of angles.

Waaa eee

Turn to **aaa** degrees azimuth and **eee** degrees elevation, where **aaa** is three digits between “000” and “360 or 450: vary according to controller type,” and **eee** is three digits between “000” and “180.” Rotators respond immediately.

Wsss aaa eee aaa sss ...

This command is similar to the [M] command: the first parameter is a time interval, and succeeding parameters are angles. With this command, however, angles are in azimuth-elevation pairs, each pair representing one antenna location. At most “1900” pairs can be sent and stored in the Control Interface. As with the other commands, the time interval range is limited to “001” to “999” (seconds), azimuth to “000” to “360 or 450: vary according to controller type,” and elevation to “000” to “180.”

When this command is sent, the rotators turn to the first **aaa** azimuth parameter and the first **eee** elevation parameter, and wait for a subsequent [T] command to begin the actual stepping (to the next azimuth-elevation pair). Stored values remain in effect until another [W] command is issued (this may have no parameters, in which case the “? >” error prompt is returned, but memories are still cleared), or until the controller is turned off or by toggling the **GS-232A** off and on.

18

COMMAND LIST	
Returned by [H] Command:	Returned by [H2] Command:
----- COMMAND LIST 1 -----	----- HELP COMMAND 2 -----
R Clockwise Rotation	U UP Direction Rotation
L Counter Clockwise Rotation	D DOWN Direction Rotation
A CW/CCW Rotation Stop	E UP/DOWN Direction Rotation Stop
C Antenna Direction Value	C2 Antenna Direction Value
M Antenna Direction Setting. MXXX	W Antenna Direction Setting. WXXX YYY
M Time Interval Direction Setting. MTTT XXX XXX XXX ... (TTT = Step value) (XXX = Horizontal Angle)	W Time Interval Direction Setting. WTTT XXX YYY XXX YYY ... (TTT = Step value) (XXX = Horizontal Angle) (YYY = Elevation Angle)
T Start Command in the time interval direction setting mode.	T Start Command in the time interval direction setting mode.
N Total number of setting angles in "M" mode and traced number of all datas (setting angles)	N Total number of setting angle in "W" mode and traced number of all datas (setting angles)
X1 Rotation Speed 1 (Horizontal) Low	S All Stop
X2 Rotation Speed 2 (Horizontal) Middle 1	O2 Offset Calibration
X3 Rotation Speed 3 (Horizontal) Middle 2	F2 Full Scale Calibration
X4 Rotation Speed 4 (Horizontal) High	B Elevation Antenna Direction Value
S All Stop	
O Offset Calibration	
F Full Scale Calibration	

De esta forma podemos resumir los comandos como (los importantes en negritas):

- O** - Calibración del potenciómetro interno del azimut
- O2** - Calibración del potenciómetro interno de la elevación
- F** - Calibración completa de la escala del azimut
- F2** - Calibración completa de la escala de elevación
- R** - **Empieza a mover el rotor hacia la derecha (Right)**
- U** - **Empieza a mover el rotor hacia arriba (Up)**
- L** - **Empieza a mover el rotor hacia la izquierda (Left)**
- D** - **Empieza a mover el rotor hacia abajo (Down)**
- A** - **Detiene el movimiento de azimut**
- E** - **Detiene el movimiento de elevación**
- S** - **Detiene o cancela el comando actual sin completarlo. (Stop)**
- C** - **Regresa el grado de azimut en la forma de "+0nnn" grados**

B - Regresa el grado de elevación en la forma de "+0nnn" grados

C2 - Regresa azimut y elevación: "+0aaa +0eee"

Xn - Selecciona la velocidad de giro del azimut

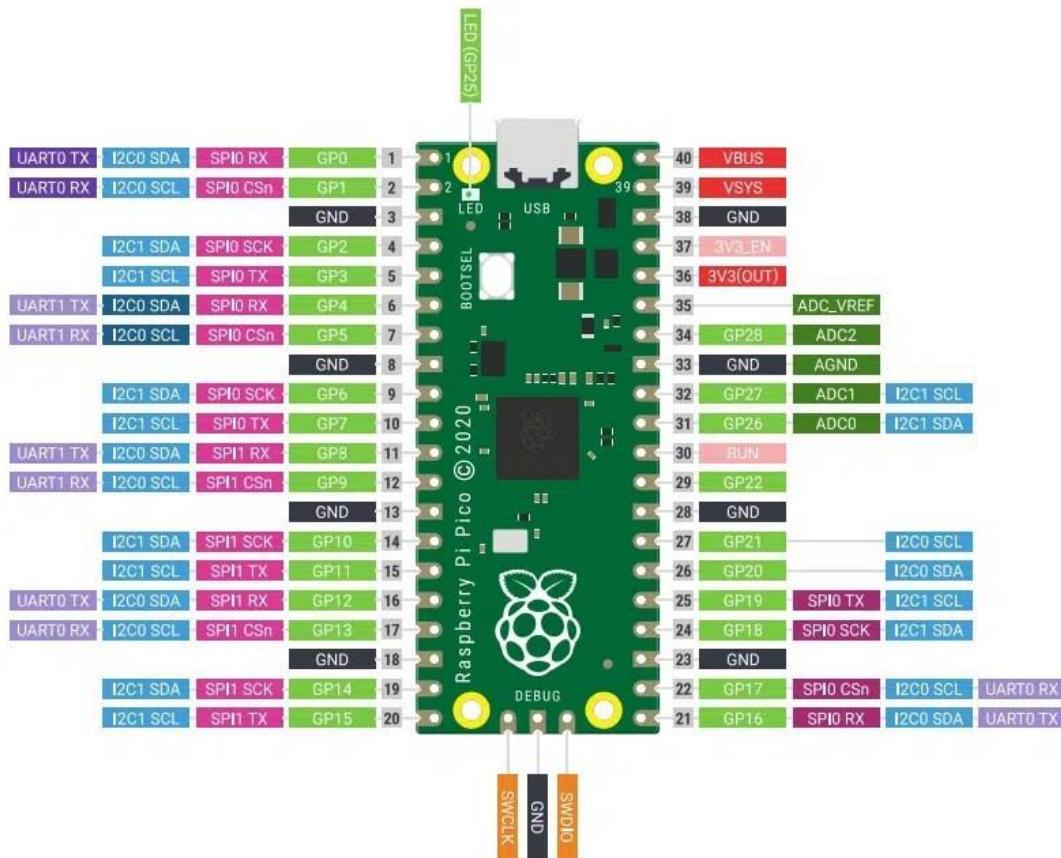
Maaa - Gira el rotor a los grado aaa

Msss aaa bbb ccc - Gira en intervalos de tiempo y conforme una lista

Waaa eee - Gira a aaa grados azimut y eee grados de elevación.

3. Arquitectura y codificación Raspberry Pi Pico

Para el proyecto se decidió usar la tarjeta del microcontrolador Raspberry Pi Pico por lo que se tuvo que investigar sobre su arquitectura, sus características y sus capacidades de programación. Se decidió utilizar ésta debido a su bajo consumo energético, su alta capacidad de procesamiento y los convertidores analógico-digitales que ya vienen integrados desde el microprocesador. Además, ya tiene integrada la comunicación serial USB que se usa para conectarse a la computadora. En la imagen se ven las distribuciones de sus pines externos.



Para el proyecto se evaluó la opción de programar en Python o en C++ y finalmente se decidió por la segunda debido a su eficiencia y el amplio catálogo de bibliotecas implementadas en este lenguaje, además de ser más flexible. Para el desarrollo se usaron las bibliotecas **stdio.h**, **string.h** y **math.h** de C para trabajar con las cadenas de comandos y las operaciones matemáticas, **pico/multicore.h** para poder utilizar los dos núcleos de forma paralela, **pico/stdlib.h**, **hardware/gpio.h** y **hardware/adc.h** para poder usar las características básicas de la Pi Pico junto con los pines y convertidores analógicos-digitales. También se usó una biblioteca compatible de Arduino llamada **liquidCrystal** para poder controlar un display LCD que le diera al usuario información extra del sistema.

El código completo se encuentra dentro del directorio Código de la carpeta donde se encontró este reporte y está comentado para poder ser revisado y comprendido, permitiendo su futuro mantenimiento. Sin embargo, hablaremos un poco sobre su estructura y lógica de procesamiento. Para empezar el programa consiste en tres archivos propios: **controlRotor.cpp** que es donde se encuentra el flujo de los dos hilos de ejecución (uno por cada núcleo), **funciones.h** que contiene las definiciones de todos los pines, las funciones auxiliares y otros parámetros, y **funciones.cpp** que contiene todas las funciones auxiliares que se usan por los hilos de ejecución principales.

El archivo **controlRotor** que es el principal, primero tiene el código del segundo núcleo. Este núcleo se encarga de estar leyendo constantemente la posición de los rotores y haciendo un promedio móvil. Después de determinar la posición actual checa si el primer núcleo le envío el código de algún comando válido o si ya se encuentra en ejecución alguno otro. Se realizan las acciones pertinentes y se muestra en el display la información correspondiente, así como también se mueve el rotor según corresponda. Este núcleo es básicamente el encargado de toda la lógica, exceptuando la lectura y procesamiento de las cadenas de los comandos. También se tiene el código main que se ejecuta en el primer núcleo y desde aquí inicia la ejecución del segundo núcleo. En el primer núcleo se está constantemente tratando de leer del buffer de entrada y se procesa la cadena para avisarle al segundo sobre el comando y los datos recibidos.

El archivo **funciones** contiene las funciones auxiliares que usa el segundo núcleo para realizar su funcionamiento. Se tiene una función de inicialización que se encarga de las primeras configuraciones del display y de los elementos de la tarjeta. Se tiene otra función llamada mapeo que convierte la lectura digital de posición en grados. Otra función para la lectura de las posiciones y realizar el promedio móvil. Una función para mover los rotores (enviar las señales correspondientes según el comando y posición deseada) y la función que dibuja en el display LCD la información correspondiente. Ya el funcionamiento y lógica de cada función se puede ver directamente en el código.

El último aspecto del código que habría que mencionar es la modificación de uno de los códigos fuente de la biblioteca de comunicación USB que viene incluida en el kit de desarrollo del software de la Pi Pico. La biblioteca se llama TinyUsb y permite el usar la Pi Pico como host o como device de una comunicación a través del cable USB que se conecta al puerto micro USB de la tarjeta. El motivo por el que se tiene que modificar este código fuente es debido a que utiliza un control de flujo que no es del todo compatible con la aplicación utilizada SatPC32. En alguna sección existe una comprobación de saludo “handshake” que no se logra efectivamente, por lo que se bloque la comunicación. Hasta que este “handshake” no se lleve a cabo de forma correcta, no se cambia el estado interno de la variable del DTR (Data Terminal Ready) que indica que se puede transferir información. Dentro de las pruebas se utilizó el software PuTTY para ver este comportamiento, al abrir dicha consola se podía comunicar perfectamente. E inclusive si lo cerrábamos y abríamos el SatPC32 se podía comunicar sin problemas ya que ya había pasado la etapa del “handshake” de la comunicación. Sin embargo, si directamente se abre SatPC32 no se da la comunicación ya que el software no manda las señales correspondientes para el “handshake”.

Lo ideal sería el poder identificar dentro de la biblioteca de TinyUsb device dónde se lleva a cabo la comprobación del saludo y bríncarsela directamente por software en el código. Sin embargo, el flujo del código no está muy claro y tiene pocos comentarios. La función de impresión de bitácora no puede activarse sencillamente desde la Pi Pico, por lo que el rastrear cuándo y dónde se produce el saludo es complicado. La solución a la que se llegó fue el forzar directamente el éxito en la comprobación del bit DTR del status de la línea de comunicación. Para esto se debe modificar: **pico-sdk\lib\tinyusb\src\class\cdc\cdc_device.c**

Cambiar la línea **391** de :

```
p_cdc->line_state = (uint8_t) request->wValue;
```

a:

```
p_cdc->line_state = ((uint8_t) request->wValue) | 1;
```

Con esto ya se puede realizar la comunicación en el momento en que se conecta el cable USB y la Pi Pico detecta una alimentación. El punto negativo de esto y por lo que se desea en una futura versión cambiar el código, es que esto hace que la Pi Pico vea como si la conexión estuviera activa desde el momento en que se conecta a la computadora y no sólo cuando se abre una aplicación que quiera usar el canal de comunicación. Esto no resulta en un problema de funcionamiento del código, pero sí limita el poder informarle al usuario final si hay un programa que está utilizando el canal de comunicación o sólo está conectado el cable USB a la computadora.

4. Configuración de RP Pico

Para empezar a programar la Raspberry Pi Pico primero debemos preparar nuestro ambiente, en este caso se usó el lenguaje C/C++ por lo que toda esta información se puede obtener del documento: “*Getting started with Raspberry Pi Pico. C/C++ development with Raspberry Pi Pico and other RP2040-based microcontroller boards*” que se puede obtener de la página web oficial de Raspberry Pi a través del siguiente enlace: <https://datasheets.raspberrypi.com/pico/getting-started-with-pico.pdf>.

En el caso de estar utilizando una Raspberry Pi 4B o Raspberry Pi 400 como plataforma de desarrollo se puede hacer una instalación muy sencilla. Tan solo es necesario el correr los siguientes comandos para obtener un script e instalarlo.

```
$ wget https://raw.githubusercontent.com/raspberrypi/pico-setup/master/pico_setup.sh
```

Una vez obtenido se procede a volverlo ejecutable y después ejecutarlo.

```
$ chmod +x pico_setup.sh  
$ ./pico_setup.sh
```

Finalmente se necesita reiniciar el equipo, lo que se puede hacer con el siguiente comando:

```
$ sudo reboot
```

Lo que realiza el script es:

- Crear un directorio llamado **pico** (donde se corre el script)
- Instalar las dependencias requeridas
- Descargar los repositorios **pico-sdk**, **pico-example**, **pico-extras**, y **pico-playground**
- Define variables de entorno **PICO_SDK_PATH**, **PICO_EXAMPLES_PATH**, **PICO_EXTRAS_PATH**, y **PICO_PLAYGROUND_PATH** en el archivo **~/.bashrc**
- Arma los ejemplos **blink** y **hello_world** en el directorio **pico-examples/build/blink** y **pico-examples/build/hello_world** respectivamente
- Descarga y arma **picotool** y lo copia en **/usr/local/bin**
- Descarga y arma **picoprobe**
- Descarga y compila OpenOCD (para debugging)
- Descarga e instala Visual Studio Code
- Instala las extensiones necesarias de Visual Studio Code
- Configura el UART de la Raspberry Pi para usarlo con la Raspberry Pi Pico.

Si se tiene como ambiente de desarrollo la distribución de **Linux Debian** o alguna basada en esta, también vale la pena usar el script de instalación de los pasos anteriores, ya que logrará instalar lo necesario para programar la Pi Pico de manera simple. Para otras distribuciones de Linux no se ha probado esto.

Windows 10

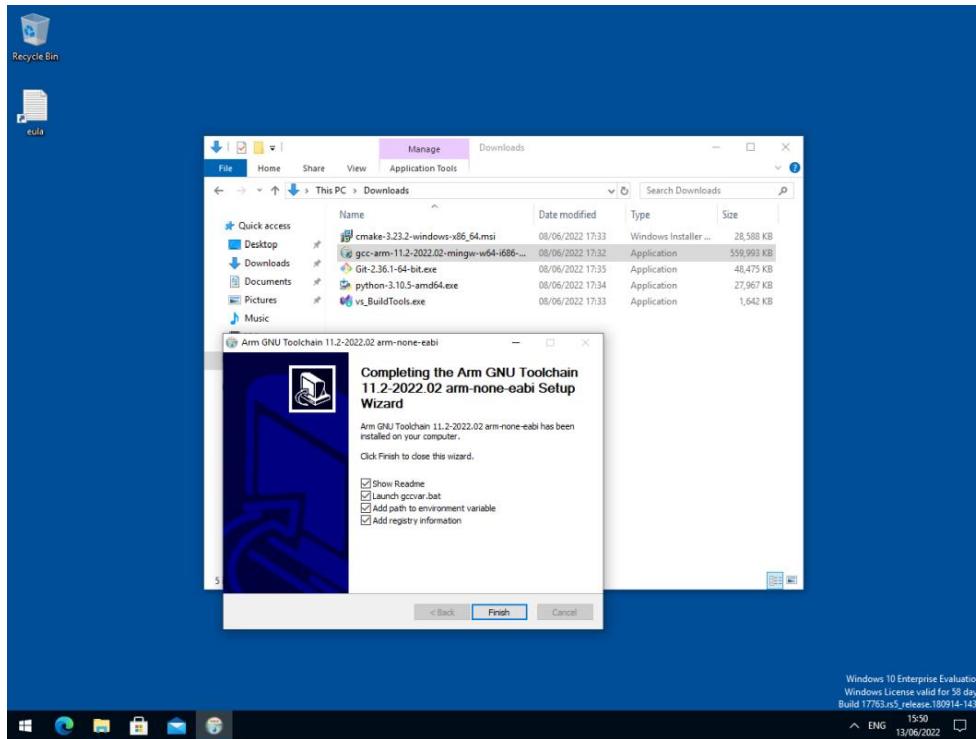
En caso de que se desee trabajar en un ambiente con el sistema operativo Windows 10 (no se soportan oficialmente Windows 7 ó 8) se necesitan realizar más pasos para la configuración del entorno que el sólo correr un script. (Se puede hacer una instalación semiautomática con un script de un tercero en el siguiente enlace: <https://github.com/ndabas/pico-setup-windows/releases> que pertenece al siguiente proyecto de github: <https://github.com/ndabas/pico-setup-windows>). Sin embargo, para la instalación manual se deben seguir los siguientes pasos. Se deben instalar las siguientes herramientas extras:

- ARM GNU Toolchain (se necesita el archivo con terminación -arm-none-eabi.exe) <https://developer.arm.com/downloads/-/arm-gnu-toolchain-downloads>
- CMake <https://cmake.org/download/>
- Build Tools for Visual Studio 2022 <https://visualstudio.microsoft.com/es/downloads/>
- Python 3.10 <https://www.python.org/downloads/windows/>
- Git <https://git-scm.com/download/win>

En las siguientes páginas se habla específicamente de cada una de estas instalaciones.

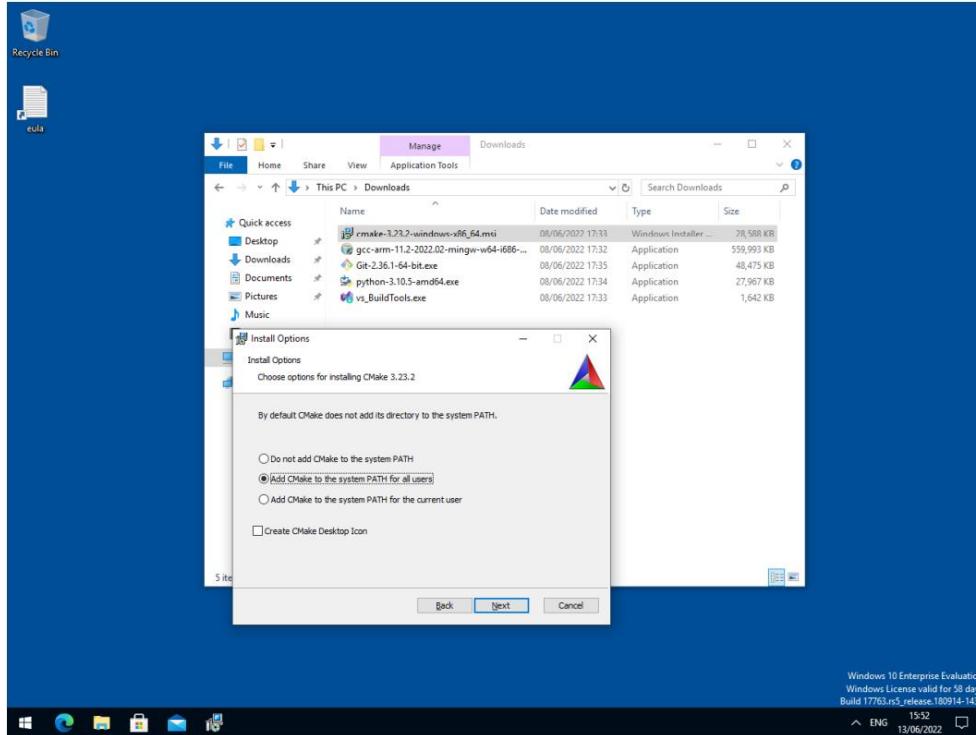
Instalación de ARM GNU Toolchain:

Durante la instalación se debe marcar la casilla “Add path to environment variable” para registrar la ruta hacia el compilador ARM como una variable de entorno en el Windows Shell.



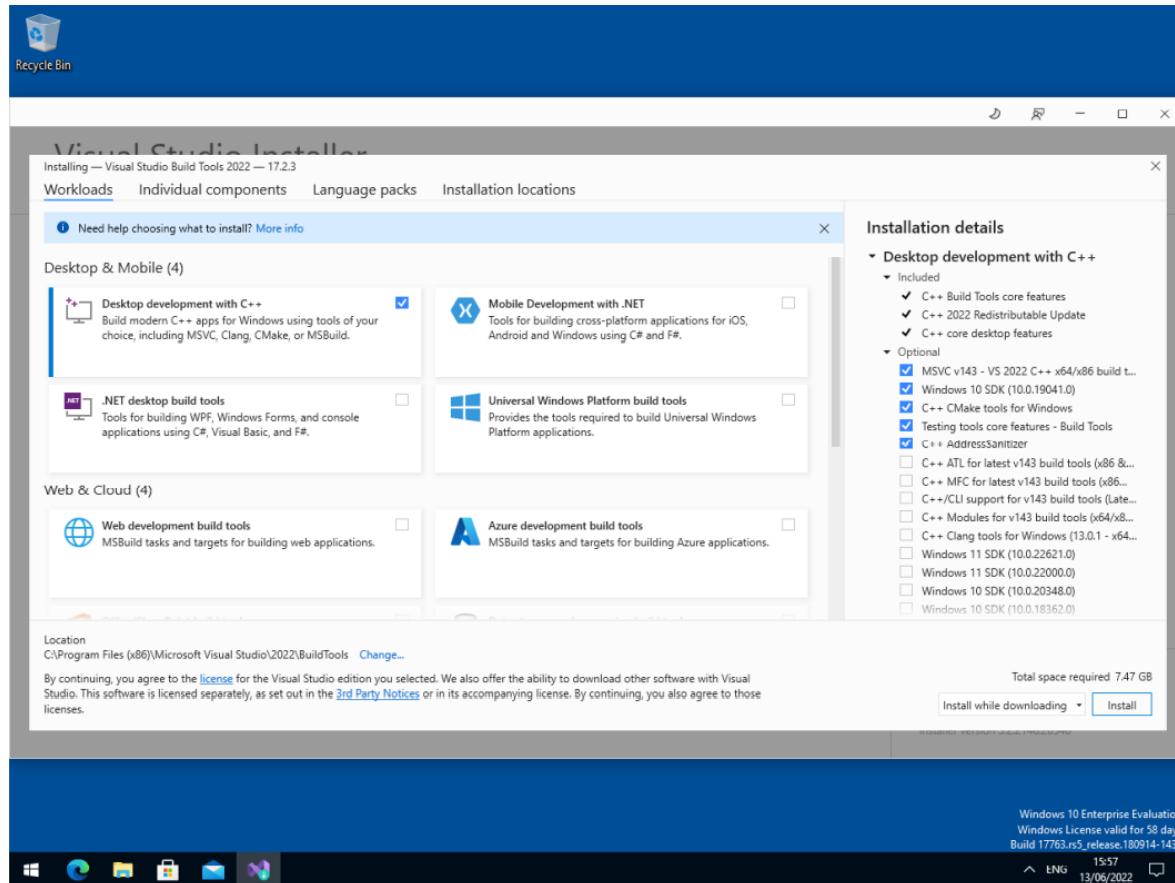
Instalación de CMake:

Durante la instalación añadir CMake a la ruta del sistema para todos los usuarios con la opción “Add CMake to the system PATH for all users”.



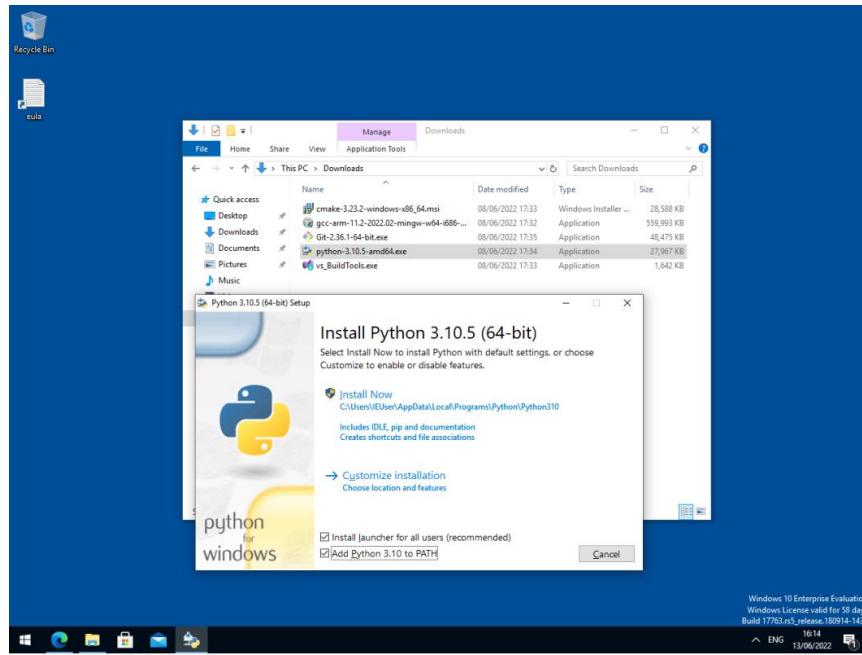
Instalación de las Build Tools for Visual Studio 2022:

Cuando el instalador solicite, se debe instalar sólo las herramientas de compilación de C++ con la opción del recuadro “Desktop development with C++”. También asegurarse que del lado derecho esté seleccionada la opción “Windows 10 SDK” porque ésta será la que arme las herramientas *pioasm* y el archivo *elf2uf2* localmente.



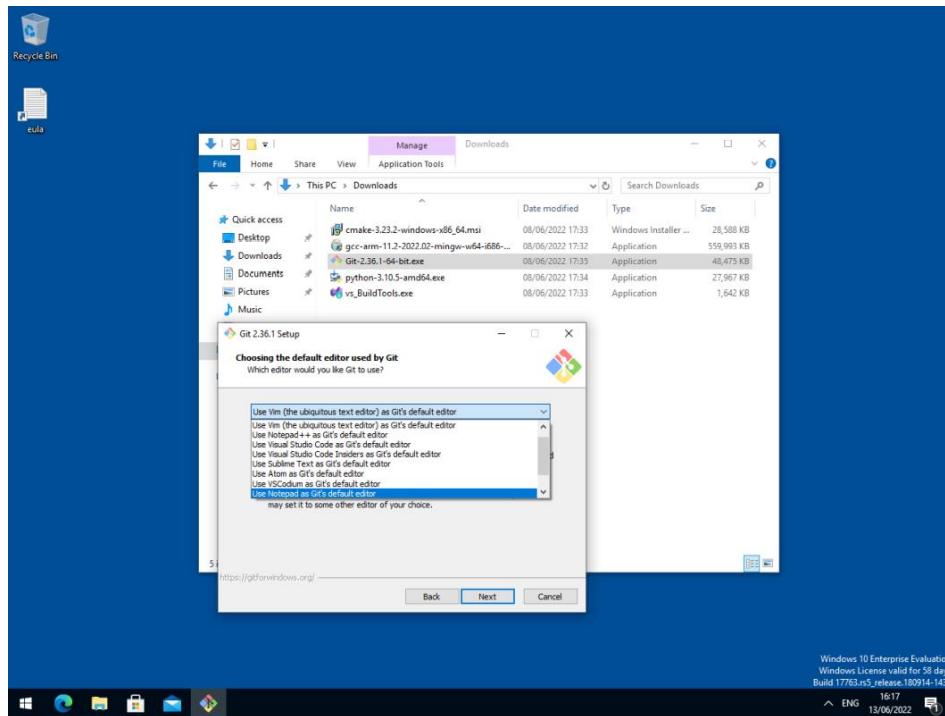
Instalación de Python 3.10:

Durante la instalación asegurarse de que esté instalado para todos los usuarios “for all users” y agregar Python 3.10 a la ruta del sistema cuando lo solicite el instalador, además se deberá deshabilitar el límite de largo *MAX_PATH* al final de la instalación de Python.



Instalación de Git:

Al instalarlo asegurarse de cambiar el editor por default a otro que no sea vim. Y también de marcar la casilla para permitir que Git sea usado por software de terceros. También a menos de tener una razón contraria, marcar las casillas “Checkout as is, commit as-is”, seleccionar “Use Windows’ default console window” y “Enable experimental support for pseudo consoles” durante la instalación.



Instalación del SDK y algunos ejemplos:

Para instalar el SDK (necesario para usar las bibliotecas de la Pi Pico) se debe clonar un directorio de git con las siguientes instrucciones:

```
C:\Users\pico\Downloads> git clone -b master https://github.com/raspberrypi/pico-sdk.git  
C:\Users\pico\Downloads> cd pico-sdk  
C:\Users\pico\Downloads\pico-sdk> git submodule update --init
```

Después en caso de querer instalar los ejemplos se puede hacer de esta forma:

```
C:\Users\pico\Downloads\pico-sdk> cd ..  
C:\Users\pico\Downloads> git clone -b master https://github.com/raspberrypi/pico-examples.git
```

Compilación del código:

Una vez se tiene el entorno con todas las herramientas configuradas ya podemos compilar nuestro código personal para la Pi Pico. Hay que abrir el **Developer Command Prompt for VS 2022** desde el menú de Windows.

Una vez abierto este prompt se debe establecer la variable de la ruta hacia el SDK de la siguiente forma:

```
C:\Users\pico\Downloads> setx PICO_SDK_PATH "..\..\pico-sdk"
```

Ya que se estableció esta ruta hay que cerrar el prompt y volverlo a abrir para que esté correctamente definido. Es importante que la ruta sí lleve a donde se descargó el sdk.

Ahora se tiene que ir hacia donde se tiene la carpeta del código, crear en el mismo nivel de jerarquía un directorio llamado **build**, moverse dentro de éste y desde ahí mandar los comandos cmake y nmake para realizar la compilación. Como ejemplo se tiene el siguiente caso:

```
C:\Users\pico\Downloads> cd pico-examples  
C:\Users\pico\Downloads\pico-examples> mkdir build
```

```
C:\Users\pico\Downloads\pico-examples> cd build  
C:\Users\pico\Downloads\pico-examples\build> cmake -G "NMake Makefiles" ..  
C:\Users\pico\Downloads\pico-examples\build> nmake
```

Esto armará el archive objetivo, produciendo los archivos objetivos **elf**, **bin** y **uf2**, los cuales se encuentran dentro de cada proyecto del directorio **build**. Los archivos binarios UF2 pueden ser arrastrados y soltados directamente en la tarjeta con el microcontrolador RP2040 conectada a la computadora a través del USB.

Cargar el programa a la Raspberry Pi Pico:

1. Mientras se aprieta el botón de BOOTSEL, conectar la PICO a través del puerto USB a la computadora.
2. Ahora la Pico aparecerá como una unidad de almacenamiento en el navegador de archivos.
3. Arrastra y suelta el **archivo.uf2** a la Pico como si estuvieras copiando o moviendo un archivo.
4. La Pico se reiniciará automáticamente y comenzará la ejecución del programa cargado.

**** para más información revisar el documento “Getting Started with Raspberry Pi Pico” <https://datasheets.raspberrypi.com/pico/getting-started-with-pico.pdf>**

Agregando librerías compatibles de Arduino:

Esto lo realizaremos para poder usar la biblioteca controladora del LCD de 16x2 y no tener que llevar a cabo nosotros mismos la programación de control. El primer paso es clonar el repositorio de librerías compatibles con el siguiente comando:

```
git clone https://github.com/fhdm-dev/pico-arduino-compat.git
```

Y posteriormente inicializar el submódulo con los siguientes comandos:

```
cd pico-arduino-compat  
git submodule update --init arduino-compat/arduino-pico
```

Ahora se tiene que inicializar la biblioteca específica que se desea utilizar, en nuestro caso sería la de ***liquidCrystal***. Por lo que se necesitan los siguientes comandos:

```
cd libs/liquidcrystal
```

En Linux se debe ejecutar el siguiente comando:

```
./init.sh
```

En Windows hay que buscar el ejecutable de la carpeta y darle doble click.

Para añadir la biblioteca a nuestro proyecto se debió modificar el archivo CMakeLists.txt con la siguiente línea:

```
add_subdirectory(<ruta_hacia_liquidcrystal> build-pac-liquidcrystal)
```

Y se debe añadir dentro de la instrucción ***target_link_libraries*** la biblioteca ***pac-liquidcrystal***.

```
target_link_libraries(controlRotor
    <otras>
    pac-liquidcrystal
)
```

*** Más información en: <https://www.hackster.io/fhdm-dev/use-arduino-libraries-with-the-raspberry-pi-pico-c-c-sdk-eff55c>

5. Compilar y cargar el programa

Si sólo se quisiera subir el programa a la Raspberry Pi Pico y ya se tiene el archivo **controlRotor.uf2** de la carpeta **build**, no se necesita ninguna instalación previa, sólo arrastrarlo a la Pico cuando se monte como unidad de almacenamiento externo según las siguientes instrucciones:

Cargar el programa a la Raspberry Pi Pico:

1. Mientras se aprieta el botón de BOOTSEL, conectar la PICO a través del puerto USB a la computadora.
2. Ahora la Pico aparecerá como una unidad de almacenamiento en el navegador de archivos.
3. Arrastra y suelta el **controlRotor.uf2** a la Pico como si estuvieras copiando o moviendo un archivo.
4. La Pico se reiniciará automáticamente y comenzará la ejecución del programa cargado.

Por otro lado, si no se tiene el archivo **controlRotor.uf2** se necesitará compilar el proyecto y posteriormente cargarlo como se indica en las instrucciones anteriores. Para poder compilarlo se necesita seguir las instrucciones del capítulo **4. Configuración de RP Pico** del documento actual hasta la subsección de **Instalación del SDK y algunos ejemplos** sin realizar la parte de **compilación del código**. Una vez habiendo instalado el SDK se pueden seguir los siguientes pasos para lograr compilar el proyecto. Suponiendo que la carpeta se guardó en Documentos (sino se tendrá que modificar las rutas).

Hay que abrir el **Developer Command Prompt for VS 2022** desde el menú de Windows. Una vez abierto este prompt se debe establecer la variable de la ruta hacia el SDK de la siguiente forma:

```
C:\Users\pico\Downloads> setx PICO_SDK_PATH "..\..\pico-sdk"
```

Ya que se estableció esta ruta hay que cerrar el prompt y volverlo a abrir para que esté correctamente definido. Es importante que la ruta sí lleve a donde se descargó el sdk.

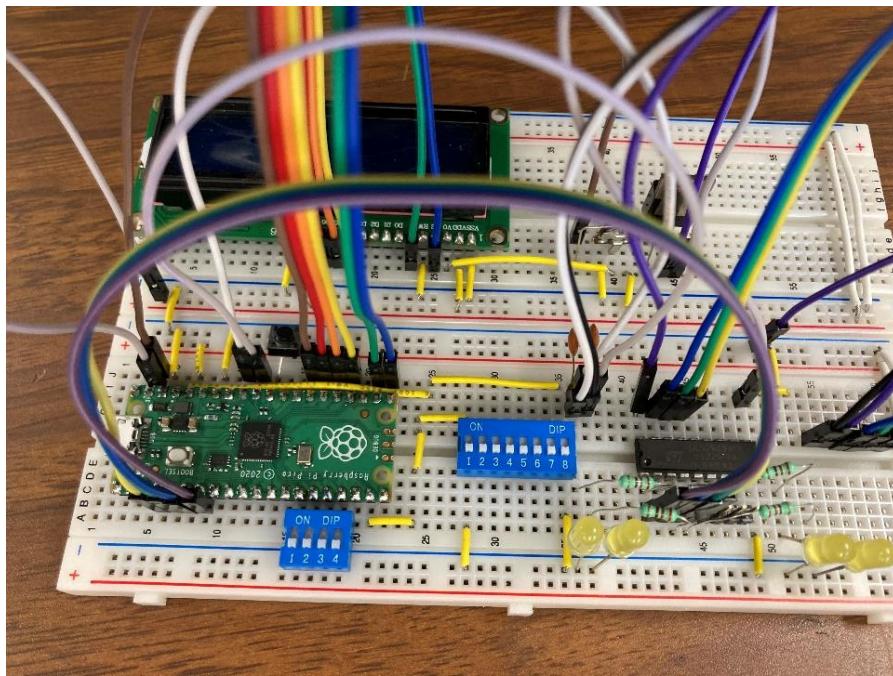
Ahora se tiene que ir hacia donde se tiene la carpeta del código, crear en el mismo nivel de jerarquía un directorio llamado **build**, moverse dentro de este y desde ahí mandar los comandos cmake y nmake para realizar la compilación. Como ejemplo se tiene el siguiente caso:

```
C:\Users\pico\Documents> cd Entregable
C:\Users\pico\Documents\Entregable> cd Código
C:\Users\pico\Documents\Entregable\Código> mkdir build
C:\Users\pico\Documents\Entregable\Código> cd build
C:\Users\pico\Documents\Entregable\Código\build> cmake -G "NMake Makefiles" ..
C:\Users\pico\Documents\Entregable\Código\build> nmake
```

Esto produce los archivos objetivos **elf**, **bin** y **uf2**, los cuales se encuentran dentro de cada proyecto del directorio **build**. Los archivos binarios UF2 pueden ser arrastrados y soltados directamente en la tarjeta con el microcontrolador RP2040 conectada a la computadora a través del USB.

6. Prototipo

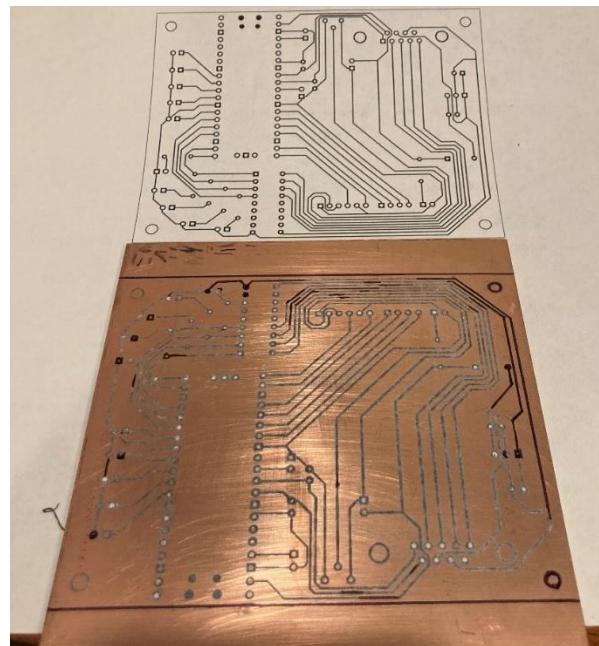
En este tipo de proyectos, el prototipado es una parte esencial del desarrollo. Conforme se iba escribiendo el código se tenían que ir probando algunas partes de su funcionamiento por separado y eventualmente todo en conjunto. Como no se sabe si va a tener que modificarse algo ya que es una etapa muy temprana del proceso, se empiezan a realizar las pruebas con ayuda de una protoboard. Ahí pudimos hacer todas las conexiones necesarias utilizando cables jumpers y también usando los componentes finales que serían soldados a la tarjeta final. En la siguiente imagen ya se puede ver el primer prototipo en protoboard a través del cual se probó el código para el display LCD, las lecturas analógico-digital del rotor y las entradas manuales de control.



De esta primera versión se modificaron varios aspectos, se agregaron resistencias, capacitores y demás elementos eléctricos para asegurar el correcto funcionamiento del circuito. Una vez ya se tenía un prototipo funcional en protoboard, que contaba con todas las entradas y periféricos que se le querían poner al sistema, se pasó al diseño del PCB. Con el primer diseño aprendí bastante, no sólo a usar el software KiCad 6, sino también a acomodar las pistas sin ángulos rectos, pasando por debajo de resistencias no superficiales y demás trucos para lograr el diseño deseado. Pero aún con este primer diseño de PCB, no se manda a hacer porque sería muy costoso, por lo que se tuvo que crear de una forma manual en el laboratorio.

Para esto primero se imprime con una impresora láser en un papel especial (una especie de papel transfer) el diseño de las pistas. Esa hoja con el diseño se plancha sobre una placa con una cara de cobre hasta que el tóner se transfiera de

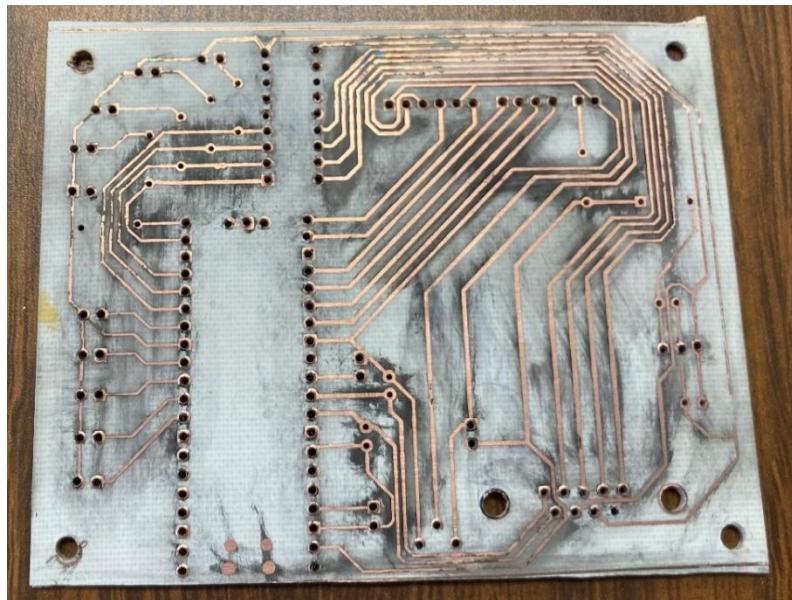
la hoja a la placa. Se retoca con un plumón indeleble las líneas que no salieron del todo bien o se cortaron y se corrobora contra el diseño de las pistas.



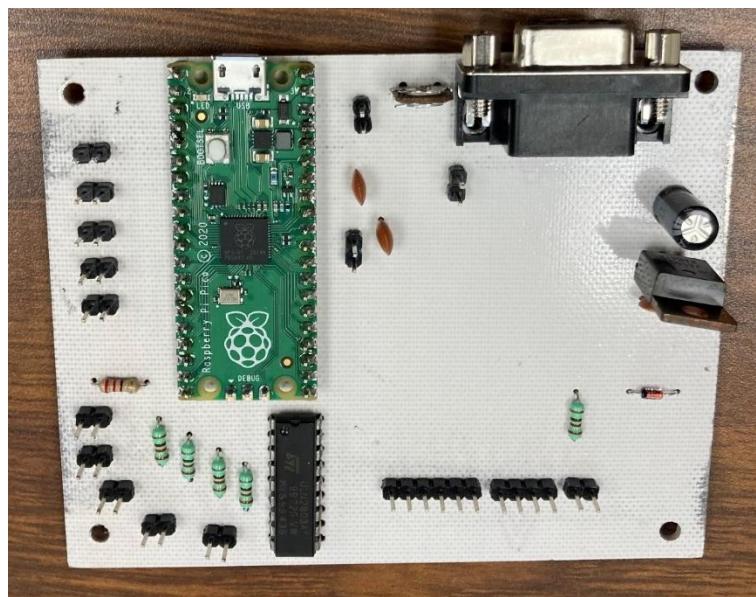
El siguiente paso es sumergirlo en una solución de cloruro férrico, el cual interactúa oxidando el cobre y sólo las pistas protegidas con el tóner o el marcador indeleble no se borran. Se deja sumergido por unos 15 minutos y luego se saca para su lavado y enjuagado. En la imagen se alcanza inclusive a ver cómo las pistas se mantienen mientras lo demás no.



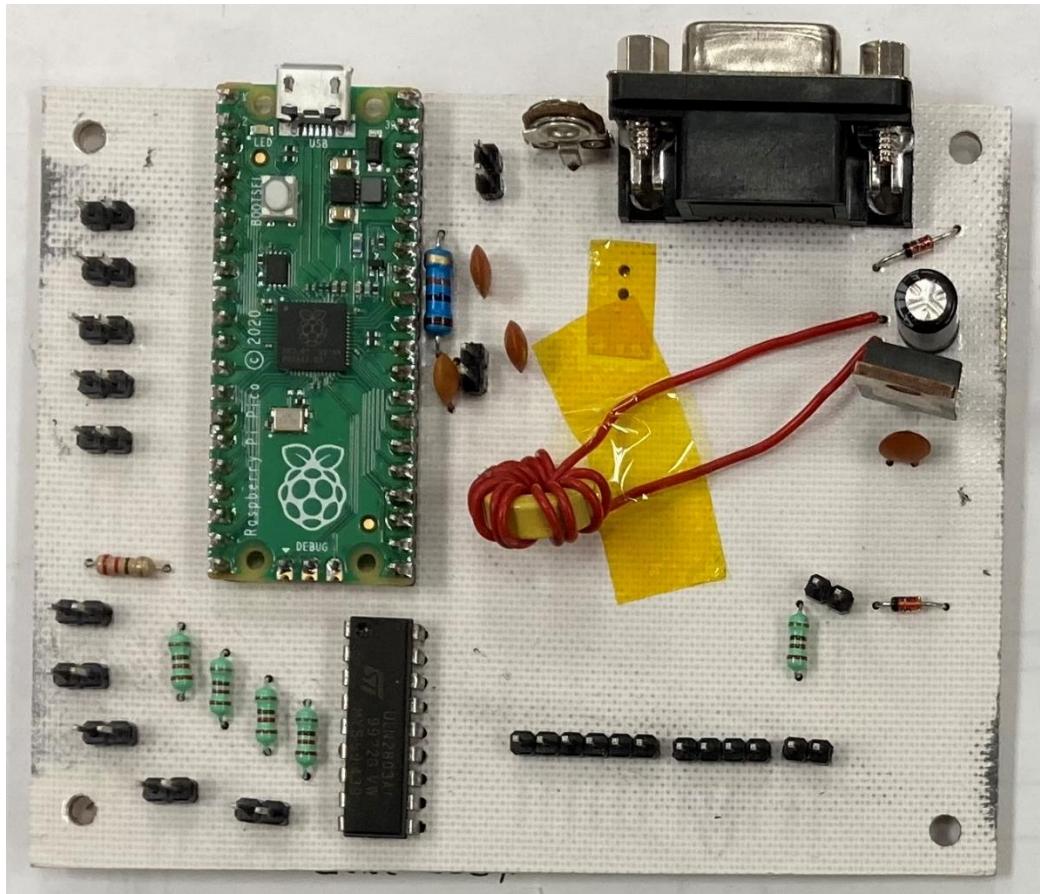
Para lavarlo y poder quitar el tóner fácilmente se utiliza acetona y ya nos quedan las pistas de la siguiente imagen. Sin embargo, en algunas secciones el plumón indeleble se borró por lo que hubo que corregir las pistas rotas con un cablecito haciendo una especie de puente. Realmente todavía en esta etapa del prototipado estos errores no representan mayores problemas y se resolvieron con pocas complicaciones. El proceso de taladrar todos los hoyitos de diferentes diámetros fue algo tedioso, pero también salió bien. Las pistas y la tarjeta prototipo ya taladrada quedaron de la siguiente forma:



Checando que con las correcciones hubiera continuidad donde debería haberla, se pasó a soldar los elementos a la tarjeta. Es importante recordar que las pistas irían en la parte trasera de la tarjeta y los elementos en la frontal para darle una mejor apariencia.

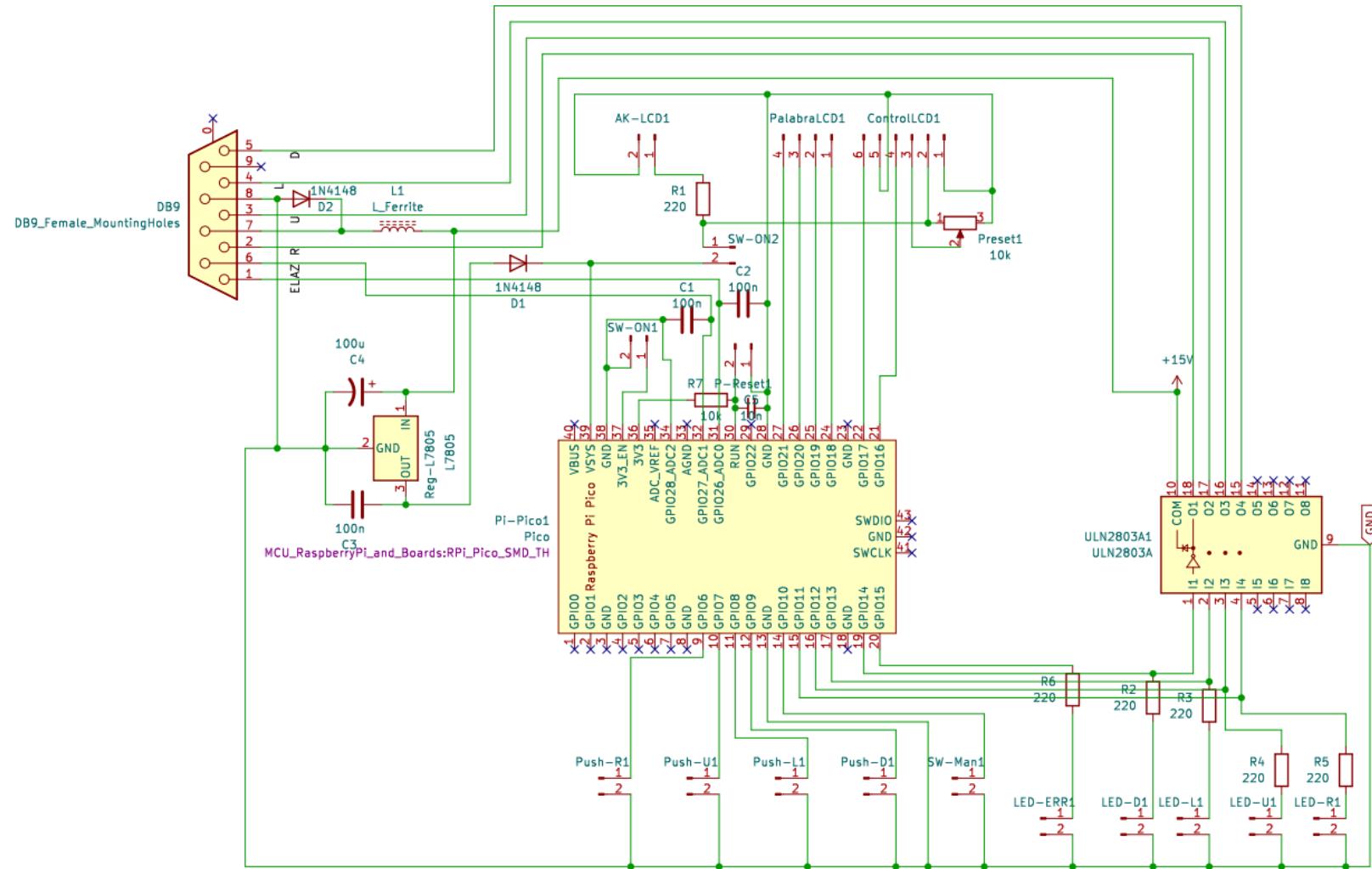


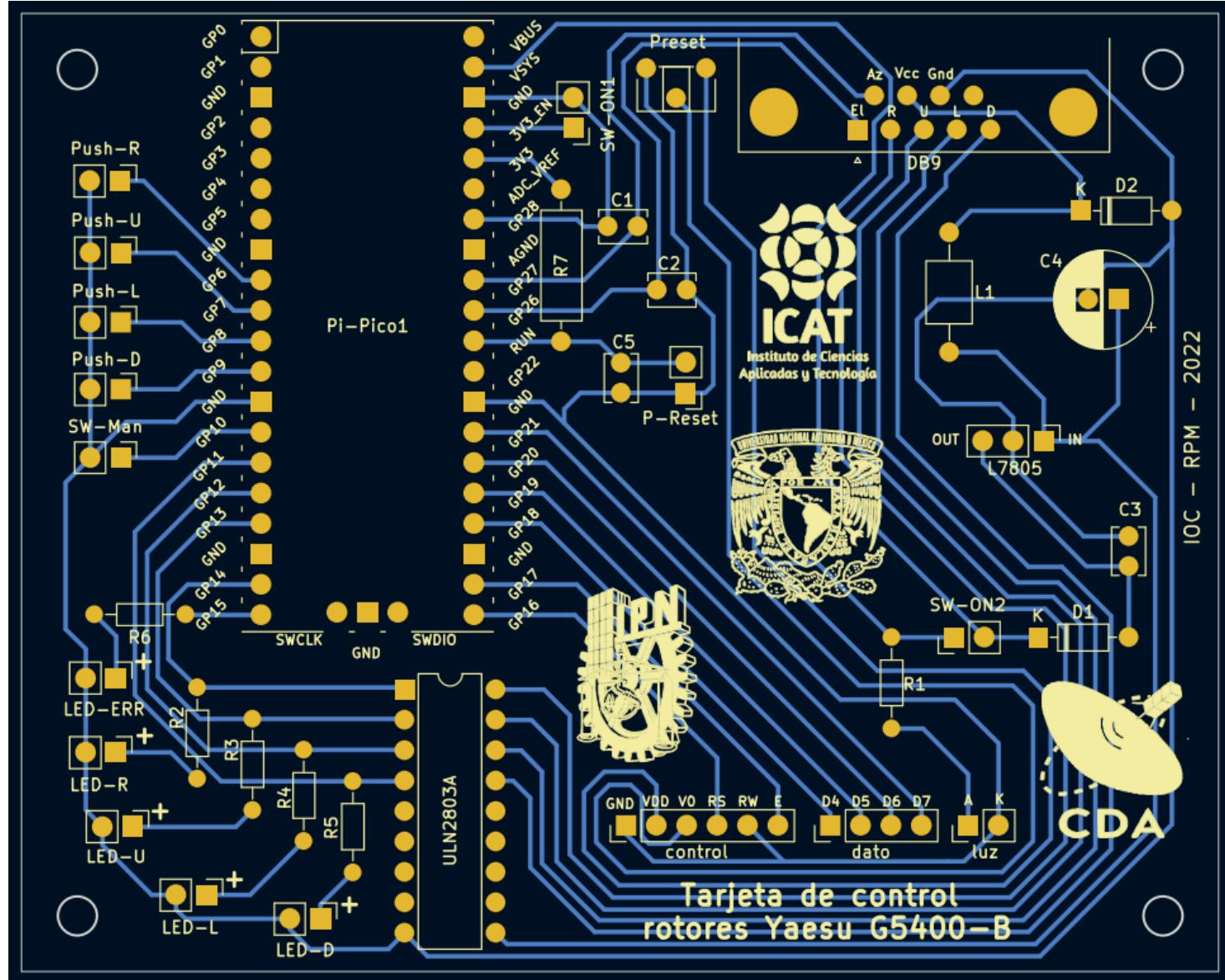
Al momento de armarla nos pudimos dar cuenta de algunos detalles que deberían cambiarse para el diseño final. Por ejemplo, se debería dar más espacio entre el regulador 7805 y el capacitor electrolítico. Además, al conectarla y probarla de repente se reiniciaba debido a una interferencia entre los cables del botón de reset y los cambios de voltaje del suministro que nos proporcionaba el rotor. Por lo que se agregó un inductor, un diodo, un capacitor y una resistencia extra para contrarrestar estos efectos. Siendo la versión final del prototipo la que se muestra a continuación. En la parte de atrás se tuvo que cortar algunas pistas y crear puentes entre otras para poder cambiar nuestro diseño y agregar los elementos necesarios.



Realmente ya con este prototipo funcional, sólo quedaba enfocarse en mejorar el diseño del PCB para que pudiera funcionar de forma óptima y también tener la mejor estética posible. Por lo que se rediseñaron las pistas con las correcciones finales y los nuevos elementos y ya se pudo mandar a hacer profesionalmente el PCB.

7. Diagramas





8. Materiales y armado

Según las notaciones en el diagrama esquemático los materiales necesarios son:

▪ DB9	Conecotor Hembra DB9	x1
▪ D1, D2	Diodo 4148	x2
▪ L1	Inductor (chico)	x1
▪ C1, C2, C3	Capacitor cerámico 100 nF	x3
▪ C4	Capacitor electrolítico 100 uF	x1
▪ C5	Capacitor cerámico 10 nF	x1
▪ R1,2,3,4,5,6	Resistencia 220 Ω	x6
▪ R7	Resistencia 10 kΩ	x1
▪ Preset1	Resistencia variables 10 kΩ	x1
▪ ULN2803A	Circuito integrado ULN2803	x1
▪ L7805	Circuito regulador 7805	x1
▪ Pi-Pico1	Raspberry Pi Pico	x1
▪ Headers machos	43 (Pi Pico) + 38 (otros)	x81

***** los push buttons, switches y leds se conectan a través de headers, al igual que el montaje de la Pi Pico.**

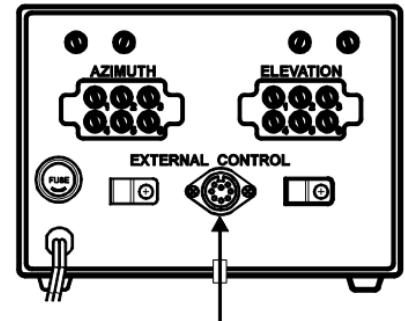
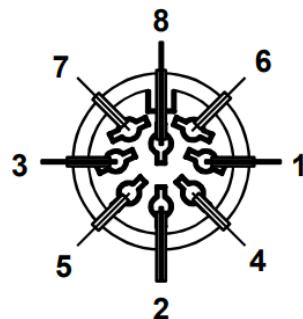
Adicionalmente como periféricos del circuito se necesitarán:

▪ Conector macho DB9	x1
▪ Conector DIM 8 pines macho	x1
▪ Display LCD 1602A	x1
▪ Push button 2 pines	x5
▪ Switch 2 polos 2 tiros	x1
▪ Switch 1 polo 1 tiro	x1
▪ Leds (4 un color + 1 otro)	x5
▪ Cable micro USB	x1
▪ Caja donde armar el circuito	x1

Lo primero que se recomienda armar es el cable de comunicación entre el control de rotores Yaesu G-5400B y el control externo que es nuestro sistema desarrollado. Para esto se utiliza el conector DB9 macho y un conector DIM de 8 pines también macho. Se deben unir mediante un cable de 8 líneas considerando la distancia que se tendrá en la instalación entre el control de rotores y el controlador externo (nuestro sistema). La forma de conectar los extremos de cada conector es según las siguientes imágenes.

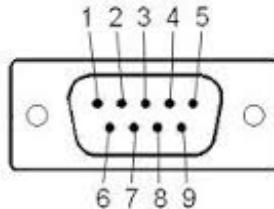
El conector DIM de 8 pines será enumerado según las especificaciones del manual:

Pin	Function
6	Provides 2 to 4.5VDC corresponding to 0 to 450°
1	Provides 2 to 4.5VDC corresponding to 0 to 180°
4	Connect to Pin 8 to rotate left (counterclockwise)
2	Connect to Pin 8 to rotate right (clockwise)
5	Connect to Pin 8 to rotate DOWN
3	Connect to Pin 8 to rotate UP
7	Provides DC13V to 6V at up to 200mA
8	Common ground



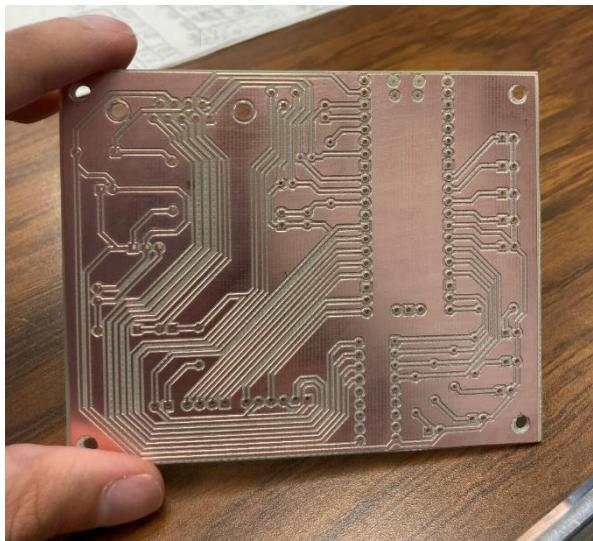
External Control

Por su parte el conector DB9 será enumerado según las convenciones como se muestra a continuación (vista de frente, es al revés cuando se solda):

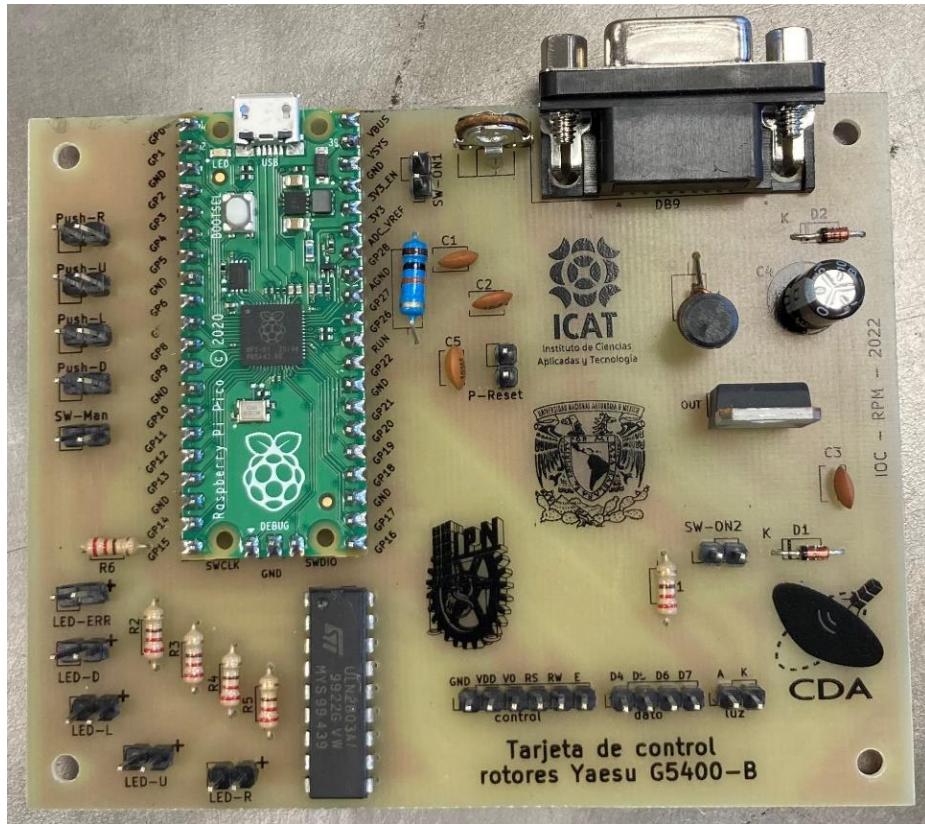


Male DB-9

El cable se debe armar uniendo las parejas de número, el pin 9 del DB9 quedará sin conectar. De esta forma ya se cuenta con el cable que conecta el control de rotores con su controlador externo (nuestro sistema). El otro cable que es un USB micro y conecta el controlador externo con la computadora deberá ser adquirido en alguna tienda contemplando la distancia entre el controlador y la computadora que tenga el software específico en la instalación final.



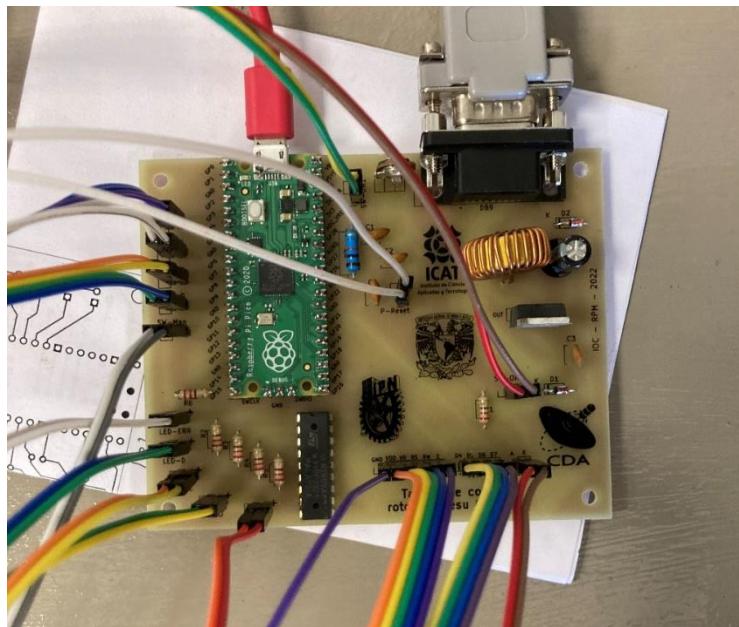
Para la tarjeta se debe mandar a hacer el circuito impreso según los diagramas mostrados en la página 26 que se pueden obtener de la carpeta PCB-ControlRotor de este entregable. Ya que se tiene el circuito impreso, el cuál es únicamente de una cara (la cara inferior), se le deben soldar todos los componentes en el lado contrario a donde se encuentran las pistas (que será el frente de la tarjeta donde se tiene la serigrafía). Terminando de soldar según los esquemas debemos obtener algo parecido a la siguiente imagen:



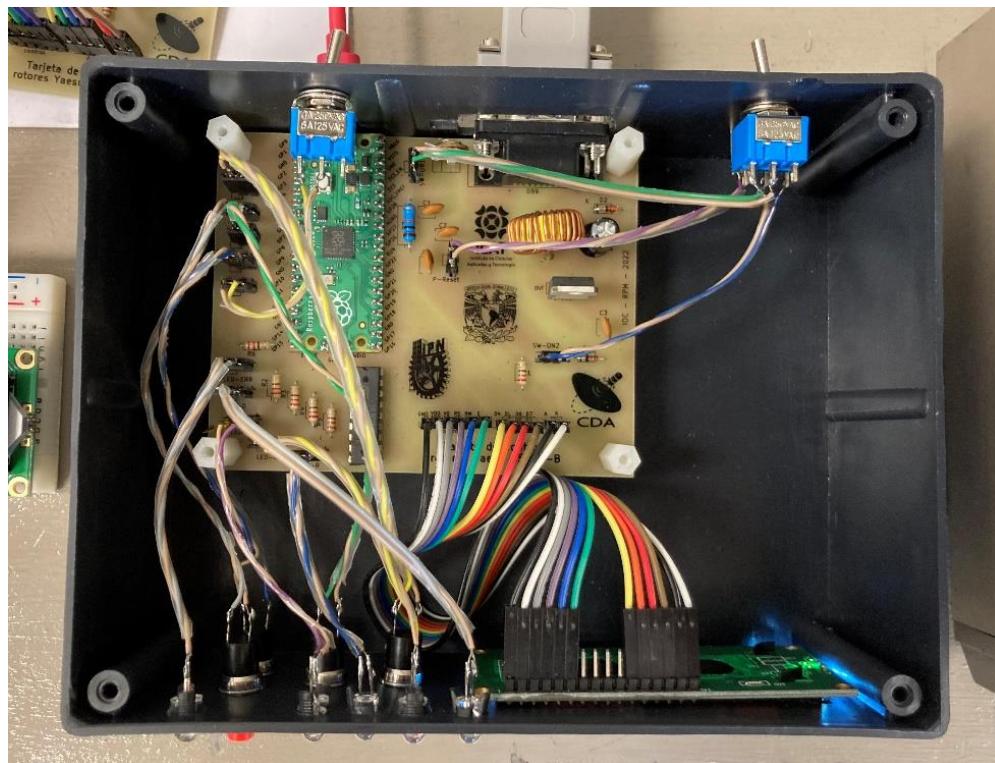
Las cuatro primeras parejas de pines del lado izquierdo superior corresponden a las señales de movimiento enviadas y se conectan a cuatro push buttons (arriba, abajo, derecha, izquierda). La quinta pareja de pines se conecta al switch manual/automático ubicado en la parte posterior de la caja de almacenamiento. Las siguientes cinco parejas de pines se conectan a los leds para mostrar el movimiento y algún posible error en los comandos ingresados. La pareja de pines en el costado superior derecho de la Pi Pico, junto con los pines a un lado del diodo D1 y la resistencia R1 se conectan al switch de dos polos dos tiros que controla el encendido y apagado del sistema. Los pines a lado del capacitor cerámico C3 se conectan a un botón de reset en la parte trasera y finalmente las hileras de pines de la parte inferior derecha se conectan al LCD.

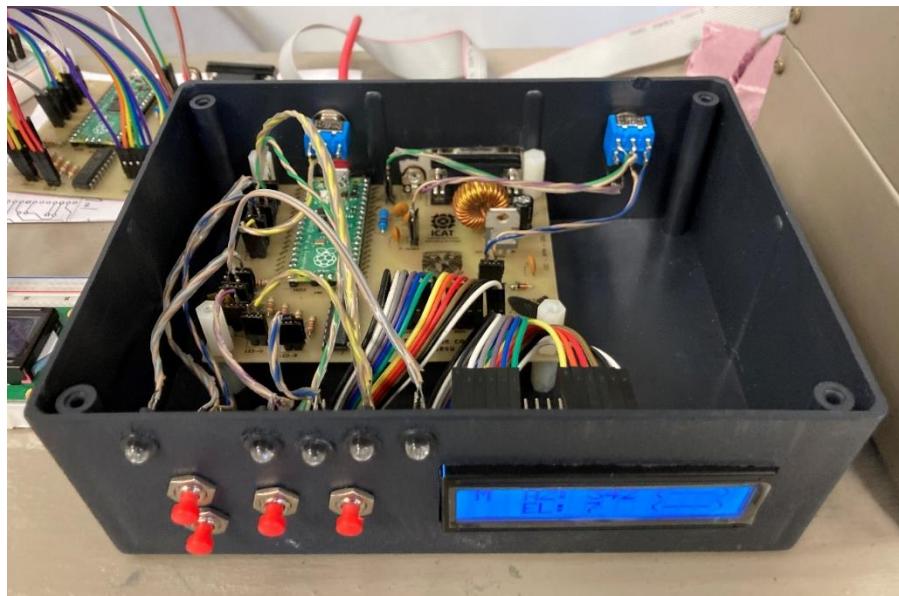
Hay que cortar los agujeros correspondientes en una caja de tal forma que en la parte frontal se tenga el display LCD, los cinco leds (cuatro movimientos y uno error) y los 4 botones de movimiento manual. En la parte posterior se necesitan los espacios para el switch manual/automático, el switch de encendido/apagado, el botón de reset, las conexiones a la computadora (micro USB) y al control de rotores (el cable entre el conector DB9 y el DIM de 8 pines). Además, se le deben ajustar los postes para fijar la tarjeta dentro de la caja con una buena altura para los conectores.

Antes de realizar todas estas conexiones y soldaduras de los cables se probó la tarjeta con los periféricos de la tarjeta prototipo fuera de la caja.



Al ver que se tenía el funcionamiento esperado ya se procedió a armar el diseño final dentro de la caja con los agujeros recortados. Como en todo momento se quiso que se pudiera armar y desarmar de forma fácil, las conexiones no son fijas y se pueden quitar. El resultado final lo podemos ver en las siguientes imágenes:





9. Configuraciones de Software



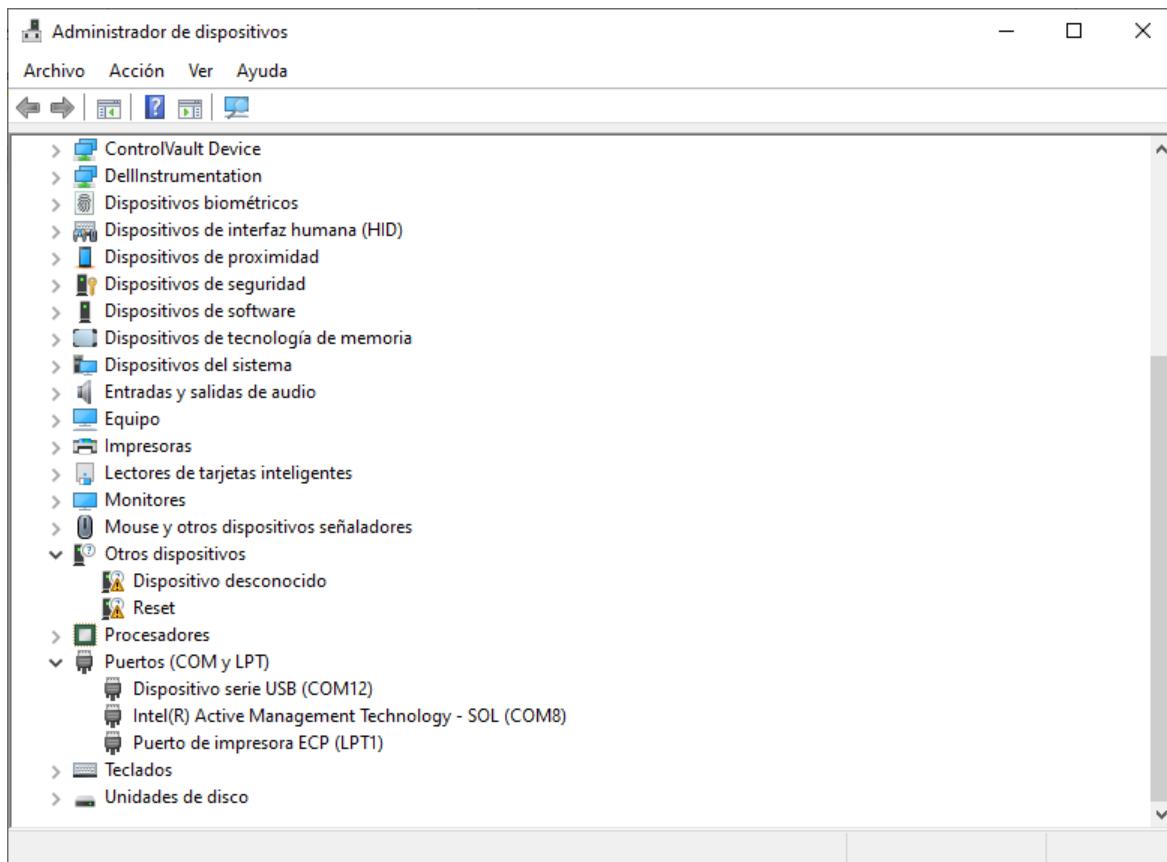
Como se mencionó anteriormente se decidió utilizar el software SatPC32 para Windows, pero para poder utilizarlo de forma exitosa se deben realizar algunas configuraciones en la computadora y su instalación. Éstas se explican a detalle en el capítulo actual.

Configuración Puerto COM Pi Pico:

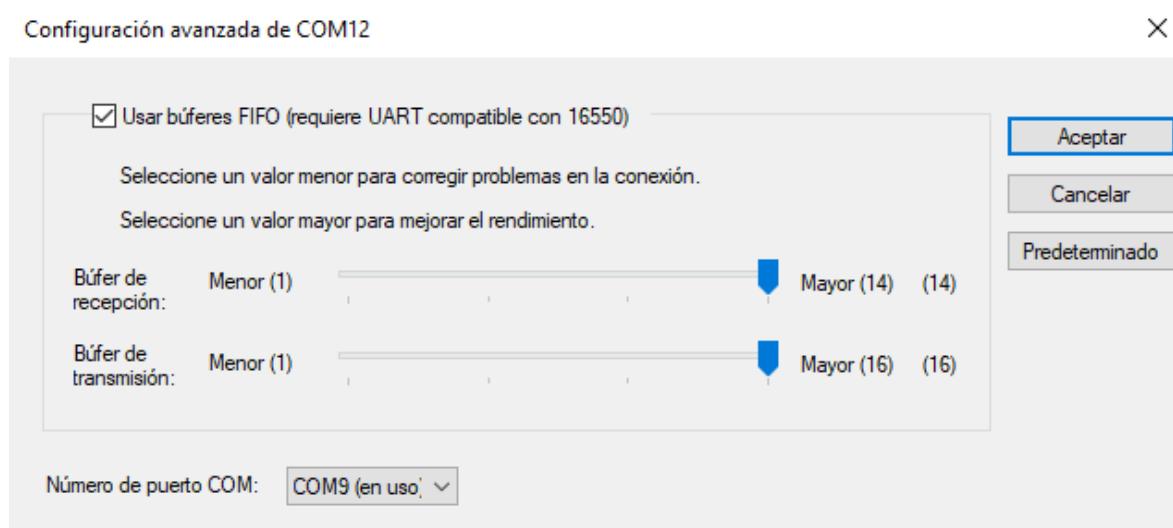
Primero una vez cargado el programa a la Pi Pico, se le da click derecho en el logo de Windows en la parte inferior izquierda de la pantalla para que se despliegue el siguiente menú.



Se selecciona la opción de **Administrador de dispositivos** y se abrirá una nueva ventana. En esta hay que ir hasta abajo y seleccionar la flechita que dice **Puertos (COM y LPT)**.



Dentro de los dispositivos desplegados, uno debe ser la Pi Pico. Para identificarla se puede desconectar y ver cuál desaparece para después volverla a conectar y ver cómo aparece nuevamente. Ya identificado cuál dispositivo es la Pi Pico, le damos botón derecho y seleccionamos **Propiedades**. En las pestañas de arriba picamos Configuración de Puerto y abajo apretamos en **Opciones avanzadas...**

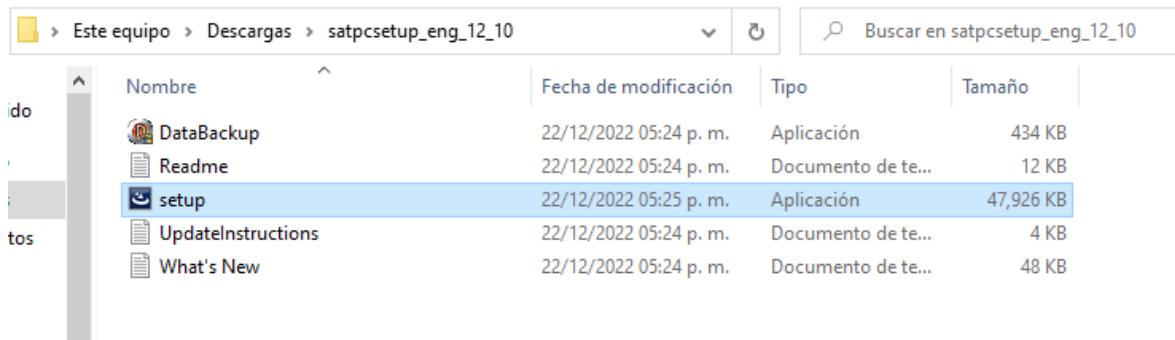


En esta pestaña hay que seleccionar algún puerto COM menor a 11, recomiendo el 9, y darle aceptar. Le damos aceptar a la advertencia que nos sale y finalmente cerramos sus propiedades dándole nuevamente aceptar. De esta forma ya queda configurada para esa Pi Pico en específico (para la tarjeta de control específica) que siempre se comunicará a través del puerto COM 9.

SATPC32 Instalación

Para descargar el programa de SATPC32 se necesita descargar el archivo SatPC32 Demo V. x.x de la siguiente liga: <http://www.dk1tb.de/downloadeng.htm>

Una vez descargada la carpeta, la descomprimimos y damos botón derecho en el archivo llamado **setup** y lo ejecutamos como administrador para poder instalarlo.



Una vez se abre el **Install Shield Wizard** le damos en Next >, nuevamente en Next y podemos llenar si queremos la información de User Name y Organization o dejarla como está y darle Next >. Recomiendo dejarle la ruta de instalación por default que es “C:\Program Files (x86)\SatPC32\” en caso de cambiársela es importante conocerla porque se necesitará más adelante. Le damos Next> y escogemos la instalación **Typical** para darle otra vez Next y finalmente Install. La instalación es bastante ligera y rápida.

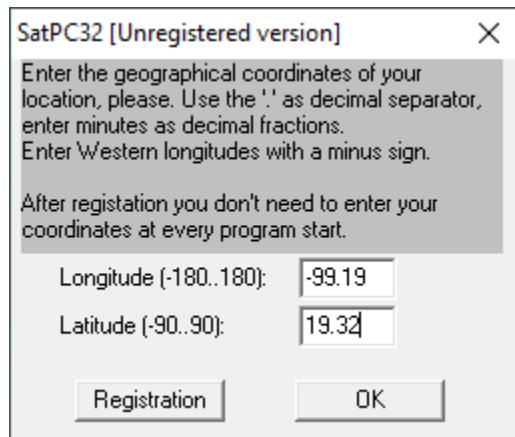
SATPC32 Configuración

Se recomienda leer el manual del programa que se encuentra aquí: http://www.dk1tb.de/manual_e.htm

Buscando en nuestro navegador de Windows “SatPC32” debe de aparecer el programa y lo seleccionamos para abrirlo. La diferencia entre la versión Demo y la de paga es que en la Demo cada vez que se inicie el programa, se deberán ingresar las coordenadas del punto donde se tiene la base terrestre (donde se encuentran las antenas por ejemplo). Las coordenadas del ICAT son aproximadamente:

Longitud: -99.18630 ≈ - 99.19

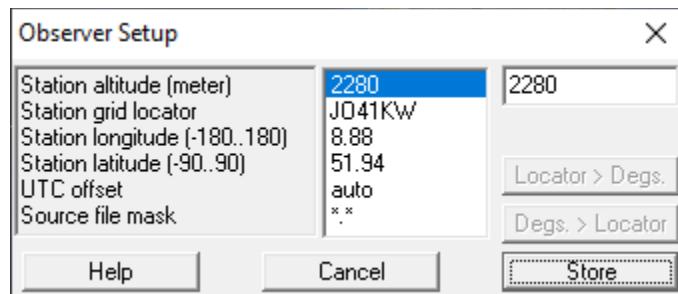
Latitud: 19.32199 ≈ 19.32



Una vez ingresadas las coordenadas le podemos dar en OK para iniciar el programa.

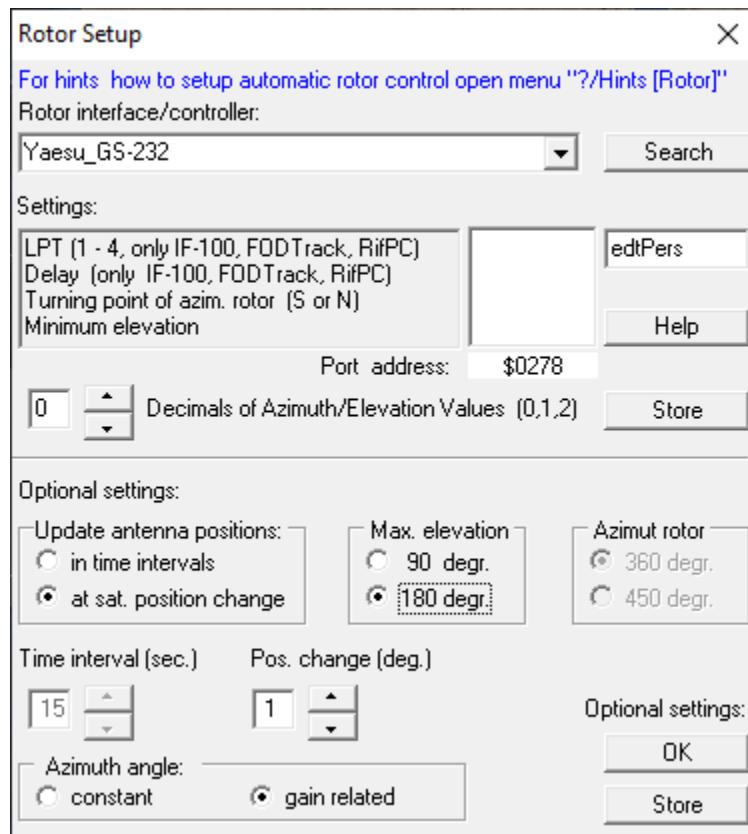
Las siguientes configuraciones son necesarias para activar el control de rotor y que esté establecido de forma correcta.

En el menú de hasta arriba en **Setup** se selecciona **Observer** y nos abre

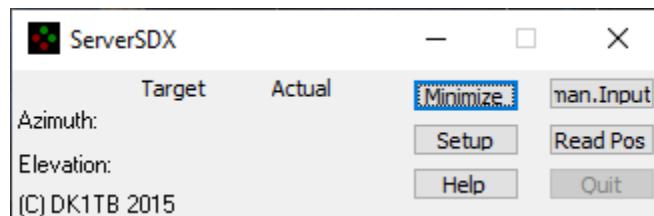


Aquí hay que modificar la altitud de nuestras antenas, en nuestro caso el ICAT está alrededor de 2280 metros sobre el nivel del mar. Lo modificamos y le damos Store.

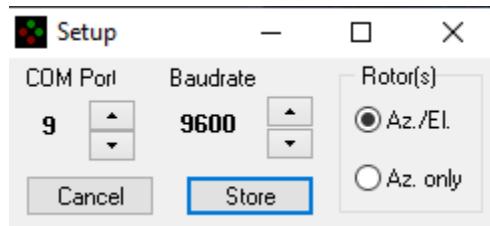
Luego nuevamente nos vamos a Setup y ahora en Rotor, aquí se debe configurar el modelo del rotor y las especificaciones del seguimiento. Para nuestro caso se imitó la interfaz Yaesu_GS-232, por lo que se selecciona esa con 0 decimales y se selecciona el Store intermedio de la pestaña (no el de hasta abajo). Nos dice que para que los cambios se apliquen hay que reiniciar el programa, pero antes también configuramos las configuraciones adicionales de forma que nos queden como en la siguiente imagen antes de darle Store hasta abajo.



Ahora reiniciamos el programa para que los cambios tomen efecto. Al iniciar nuevamente el programa se nos abrirá también una nueva pestaña:

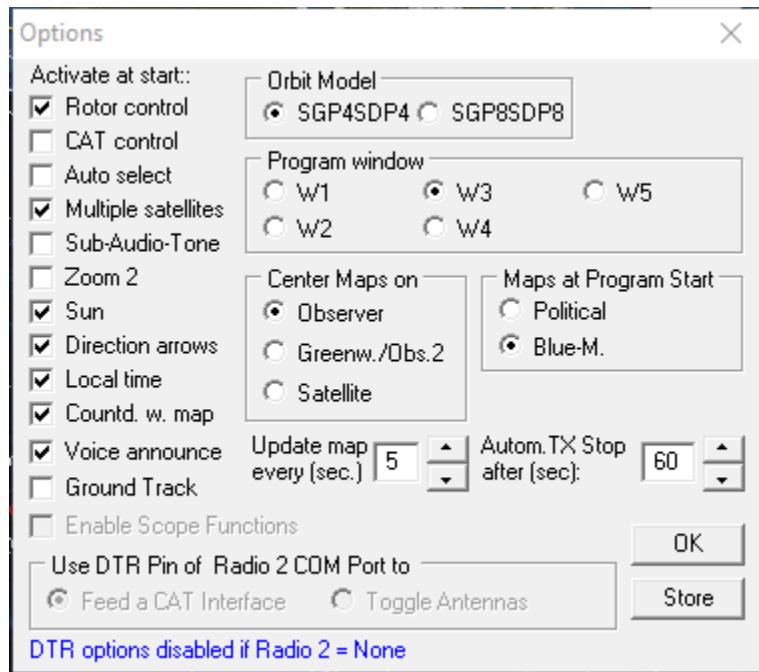


Para ver esta pestaña la podemos seleccionar de nuestras aplicaciones abiertas en la parte de debajo de nuestra pantalla de Windows. Seleccionamos en **Setup** y seleccionamos el puerto COM que ya anteriormente configuramos a la Pi Pico. En nuestro caso fue el puerto COM 9.



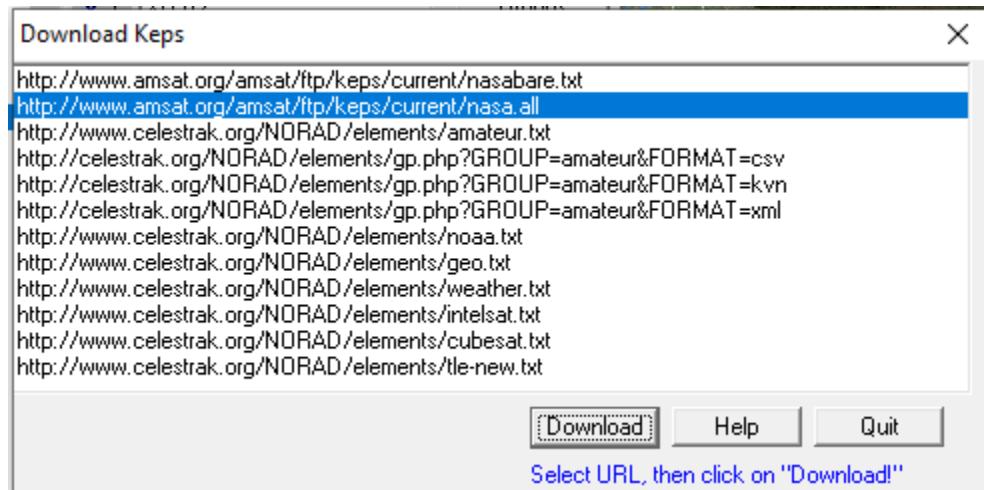
Le damos en Store y nuevamente deberemos reiniciar el programa para que los cambios tomen efecto. Cerramos el programa principal y lo volvemos a abrir.

Ya las últimas configuraciones se hacen nuevamente en **Setup** en su menú de **Options**. Las siguientes son las configuraciones recomendadas, pero se puede experimentar con ellas para dejarlas como mejor le ajuste. Lo importante es que esté activado el Rotor control y la opción de Multiple satellites para poder ver varios al mismo tiempo.

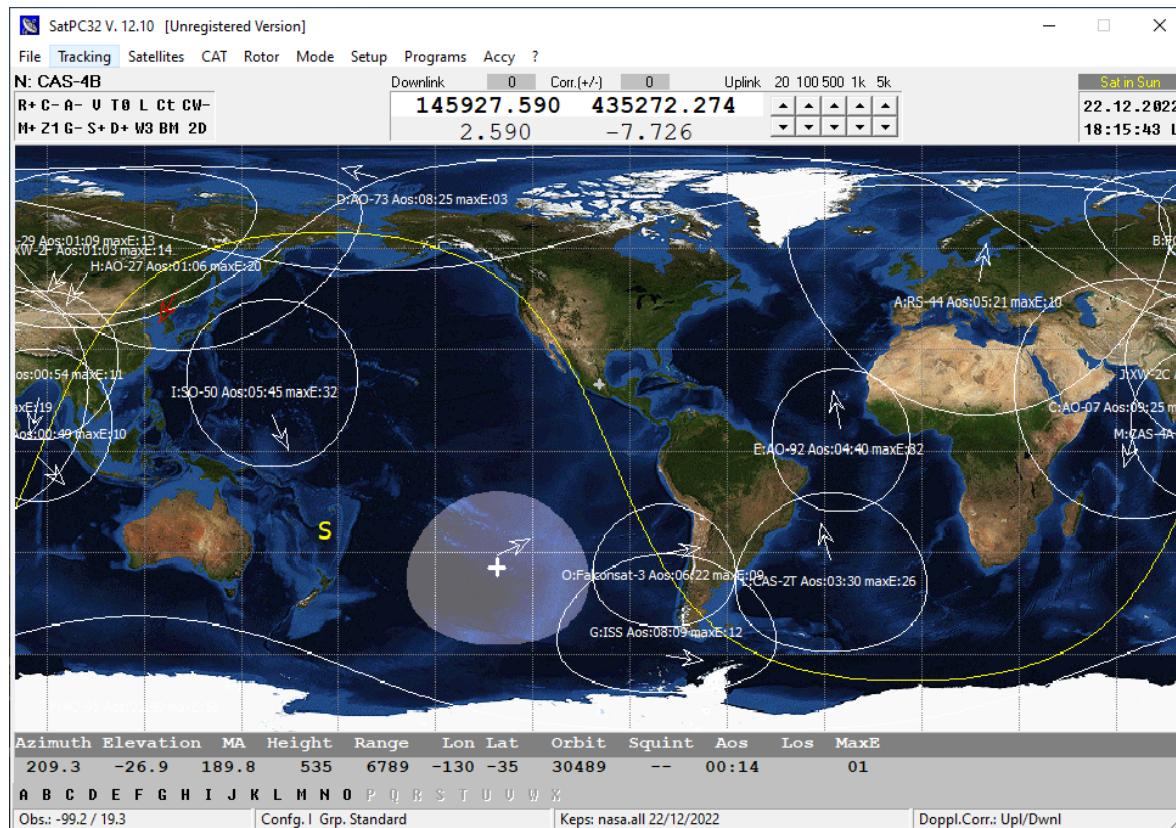


Nuevamente se necesita darle en Store y luego en Ok para guardar la configuración y finalmente reiniciar el programa para que tome efecto.

Lo último que hay que configurar son las listas de satélites que se quiere tener (hasta 26 de la A a la Z) y si se quiere que sean visibles en el mapa. Para esto en los menús superiores se selecciona **Satellites**. Seleccionamos el source file por default **nasa.all** que contiene todos los satélites que conoce la nasa públicamente. En la columna de nombres de satélites de la izquierda están todos, por lo que sólo hay que buscar el que queremos y darle doble click. Esto lo pasa a los satélites seleccionados y se le asigna la siguiente letra disponible. Lo seleccionamos en la columna de la derecha y podemos decidir si poner Show on/off para verlo en el mapa. Según queramos lo configuraremos y antes de darle Ok para salir hay que actualizar las órbitas de los satélites. Para esto en la parte inferior derecha le damos click en **Update Keps** en la ventana que se abre seleccionamos nasa.all y en Download para actualizarlos.



Finalmente le damos en la opción de Okay para cerrar la ventana. Ya con todas las configuraciones realizadas. Para seleccionar un satélite en específico en la parte de debajo de la ventana se tienen las letras que los identifican, ahí se escoge el que se quiere seguir y en cuanto su radio de visibilidad toque nuestra estación terrestre el programa en automático lo empezará a seguir.



Automatización del ingreso de coordenadas:

El mayor inconveniente del programa SatPC32 Demo es que hay que ingresar manualmente las coordenadas en cada ocasión, sin embargo, esto se puede automatizar. Para eso se crea un script batch que abra el programa e ingrese las coordenadas junto con las teclas correspondientes para ingresar. El código del **archivo.bat** es:

```
@if (@CodeSection == @Batch) @then

@echo off

set SendKeys=CScript //nologo //E:JScript "%~F0"
start "" "C:\Program Files (x86)\SatPC32\SatPC32.exe" MAX
timeout /t 5
%SendKeys% "-99.19"
%SendKeys% "{TAB}"
%SendKeys% "19.32"
%SendKeys% "{TAB}"
%SendKeys% "{ENTER}"
%SendKeys% "{ENTER}"

goto :EOF

@end
// JScript section

var WshShell = WScript.CreateObject("WScript.Shell");
WshShell.SendKeys(WScript.Arguments(0));
```

Es importante hacer notar que, si se instaló en alguna ruta que no sea la de default, se tiene que modificar la ruta que le sigue a la instrucción **start**. De igual forma si la computadora es muy lenta se podría tener que aumentar el tiempo de **timeout** que es lo que espera antes de ingresar las coordenadas y es lo que puede tardarse en abrir el programa. También se le pueden modificar las coordenadas según sea necesario.

Para crearlo y editarlo se tiene que usar un editor de códigos como Sublime Text, Visual Studio, Codeblocks, etc. Y para ejecutarlo sólo se le da doble click al archivo. Una vez se le da doble click no se debe tocar el computador hasta que abra el programa debido a que podría fallar si se le mueve algo a la pantalla o se selecciona otra ventana.

10. Modo de uso y conclusiones

El diseño final del sistema tiene las siguientes capacidades:

- Modo manual (interacción botones).
- Modo automático (comunicación con la computadora).

En ambos modos los cuatro leds se iluminan para indicar que se está enviando una señal de movimiento a los rotores. De izquierda a derecha los leds indican: subir, bajar, izquierda, derecha.

El display muestra la siguiente información:

- En la esquina izquierda superior una A si está en modo automático o una M si está en modo manual.
- En la esquina izquierda inferior muestra una U cuando recibe alimentación por el USB (está conectado a una computadora) o nada en caso contrario.
- Para las direcciones se muestra

AZ: <posición actual azimut> (<posición deseada azimut>)

EL: <posición actual elevación> (<posición deseada elevación>)

* En caso de estar en modo automático y no tener conexión USB se despliega únicamente el mensaje: Sin conexión USB...

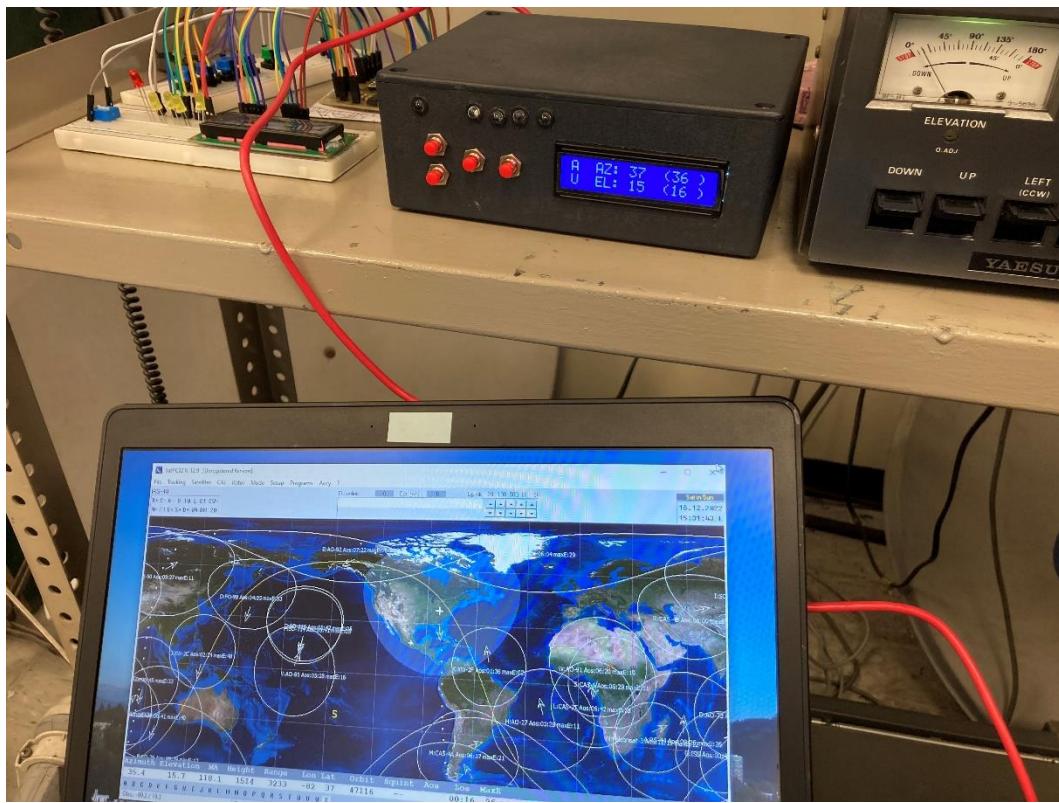


Para cambiar entre el modo manual y modo automático, se tiene un switch en la parte trasera izquierda.

Para prender y apagar el sistema (independientemente de si está alimentado por el rotor, usb o ninguno) tiene un switch en la parte trasera derecha.

Adicionalmente se cuenta con un botón de reinicio por si el sistema llegara a congelarse (no debería y no pasó en las pruebas) debajo del switch de encendido. Este botón también puede usarse para cargar un nuevo programa al sistema si se le da un doble click rápido. Se reiniciará la Pi Pico y se montará en modo almacenamiento a la computadora para poder arrastrar el nuevo programa. Si no se desea cambiar el programa se puede apagar y encender el sistema.

También se cuenta con un led rojo en la parte frontal hasta la izquierda el cual nos indica sobre errores y otros problemas. Sólo se debería prender con normalidad al encender el sistema o al montar la Pi Pico para cargarle un nuevo programa.



** En cualquier modo de uso, siempre se debe conectar el sistema al rotor mediante el cable DB9 – Dim 8 pines.

Conclusiones del proyecto

El sistema desarrollado funciona correctamente y cumple con los requerimientos que se plantearon en un inicio. La documentación del código y este mismo documento permite tanto replicar el proyecto como hacer modificaciones, mantenimiento o las mejoras correspondientes por alguna otra persona. Para futuras versiones una de las mejoras que se pudieran hacer es la separación de los pines que van conectados al display LCD, actualmente tienen una separación no estándar y sería mejor el dejarles exactamente un header de separación. Así las conexiones serían más limpias al poder usar un conector que los unifique.

En otro desarrollo se podría minimizar todavía más el espacio de la tarjeta al no usar la Raspberry Pi Pico completa, sino sólo integrar su microcontrolador RP2040 en nuestra tarjeta y así tener un diseño totalmente propio. Esto no representaría una gran dificultad ya que los diagramas esquemáticos de la Pi Pico se encuentran en el manual para que cualquiera pueda consultarlos. Por lo que sólo sería el copiar los circuitos de la tarjeta y agregar los necesarios a nuestro diseño. También se podría hacer otros códigos más eficientes que usen interrupciones para la lectura de los comandos, sin embargo, debido a los tiempos de respuesta de los rotores y del software de seguimiento utilizado, no se necesita un sistema más eficiente. Sería sólo acabar de pulirlo por gusto más que por necesidad del mismo sistema, porque ya así cumple bien y es de por sí de bajo consumo y con respuesta más rápida a la necesaria.

Conclusiones del servicio social

Más allá que el haber desarrollado un sistema que ayudará y facilitará un proyecto de una institución tecnológica pública, el servicio social me sirvió mucho para aprender acerca del desarrollo de un producto o herramienta propia. Realmente aprendí mucho más de lo que esperaba en un inicio, y en temas muy diversos que no se me hubieran ocurrido. Fue una experiencia muy agradable y con la cual siento crecí en mi desarrollo profesional, en ningún momento antes había llevado un desarrollo tan completo como el del servicio social y viendo en retrospectiva me siento satisfecho con lo realizado.

Desde el acostumbrarme a soldar manualmente los componentes, hasta el ver la importancia del prototipado y su flexibilidad en las primeras etapas de desarrollo, creo que ahora tengo una idea más completa de la creación y diseño de un producto o herramienta. Traté de aprovechar lo mayor posible la oportunidad de estar en el instituto para trabajar y aprender, a pesar de en un inicio sentirme perdido conforme iba investigando más y probando con la Raspberry Pi Pico más cómo me sentía con el desarrollo. Me gustaría en un futuro poder volver a colaborar con el ICAT y sólo me queda agradecer a los profesores del Laboratorio de Simulación y Procesos que siempre estuvieron para ayudarme a llevar con éxito el proyecto.