

MANUAL DE CONEXIÓN TXT CONTROLLER

Contenido

1. Caracterización controlador
2. Community Firmware
3. Diseño de aplicaciones
4. Librería para comunicación Paramiko
5. Sensores y actuadores
6. Troubleshooting

En el laboratorio de Ambiente Integrado de Aprendizaje se cuenta con un Centro de Simulación Multiproceso (Cesim), que está compuesto por un sistema de bandas transportadoras y dispensadores, que tiene como componentes principales controladores, sensores y motores.

1. Caracterización del Controlador

A continuación, se muestra una imagen del controlador donde se explican sus partes, así como entradas y salidas.

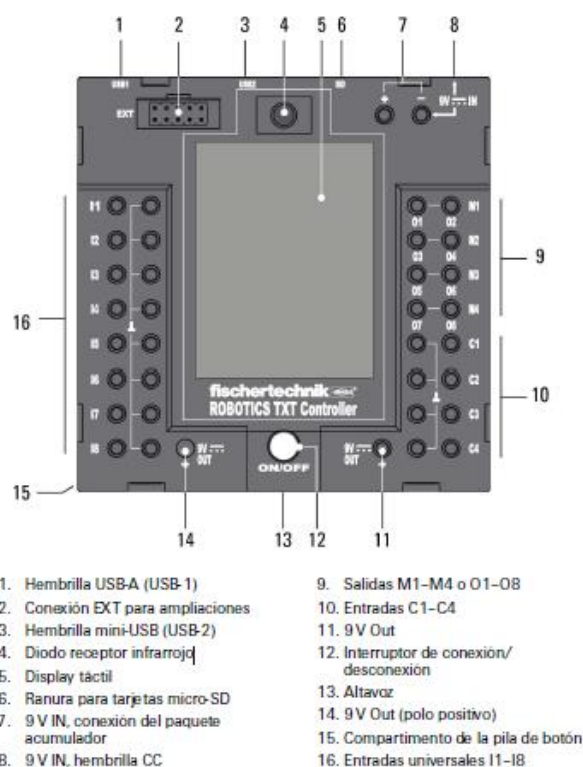


Ilustración 1 Controlador Fischertechnik

La salida que se usará principalmente según la *ilustración 1* será la número 2, la cual permite la ampliación para el uso del **protocolo I2C** para los motores. Adicionalmente, se pueden usar las salidas O1-O8 para LEDs, sensores, etc. Los valores de las salidas y entradas de voltaje serán a 9V.

La distribución del software de este controlador está construida sobre **Builtroot**, que es una distribución embebida de Linux usando compilación cruzada. Este puede soportar algunas librerías de C y por medio de elementos como el **Community Firmware** se puede correr aplicaciones de Python.

2. **Community Firmware (CF)**

A continuación, se explica el uso del **TXT controller** por medio de Python, utilizando el *Community Firmware CF* (<https://cfw.ftcommunity.de/ftcommunity-TXT/en/>).

Para el uso de este, la versión del TXT Controller tendrá que ser superior a 4.2.4. Adicionalmente, se necesita una tarjeta de memoria microSD formateada completamente (formatear usando la aplicación *SD Card Formatter*). Los pasos para la instalación del firmware son:

1. Descargar el Community Firmware (CF) de <https://github.com/ftCommunity/ftcommunity-TXT/releases/tag/v0.9.5>
2. Insertar en la micro SD **únicamente** los archivos de la carpeta comprimida descargada

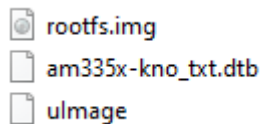
The image shows a list of three files with their respective icons: a disk icon for 'rootfs.img', a document icon for 'am335x-kno_txt.dtb', and a document icon for 'ulmage'.
3. Habilitar “Boot SD” en el TXT Controller siguiendo la ruta: *Settings > Security > Boot SD*



Ilustración 2 Habilidad Community Firmware

Luego de completar la descarga y habilitación del Community Firmware realice los siguientes pasos:

1. Apagar el TXT Controller
2. Insertar la memoria SD en el TXT Controller
3. Encender el TXT Controller

El TXT Controller deberá iniciar con el Community Firmware CF. Este tiene una apariencia diferente y más amigable con el usuario.



Ilustración 3 Interfaz con el Community Firmware

Una vez instalado el firmware, se procede a conectar el controlador a la Red de WiFi local (SSID: AIALAB, Pass: LabAIA2021).



Ilustración 4 Conexión red WiFi

Luego de conectado, presionando la parte negra superior se debe poder visualizar la dirección **IP del controlador**.



Ilustración 5 Datos del IP controlador

Ingresando a esta dirección desde un computador conectado al mismo *Access Point*, se tendrá acceso a la interfaz gráfica del Community Firmware CF. Desde esta interfaz se podrá cargar programas al TXT Controller.

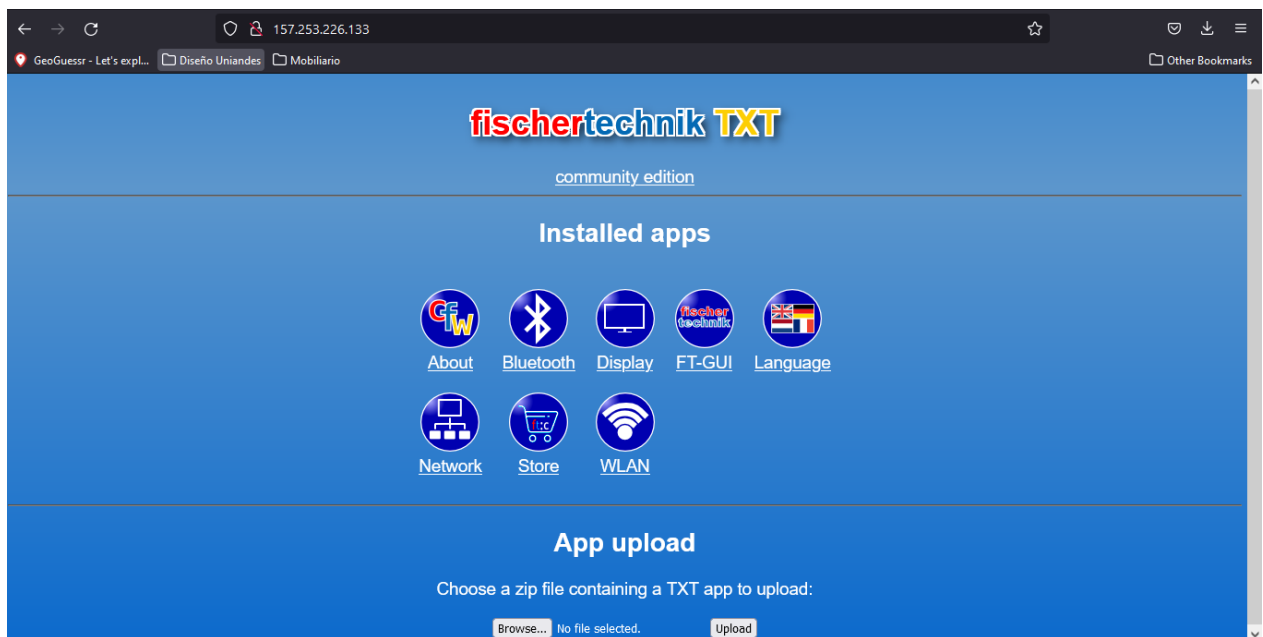


Ilustración 6 Interfaz Fischertechnik

3. Diseño de aplicaciones

➤ Estructura básica

Para el diseño de aplicaciones, se parte de la guía de aplicaciones de Community Firmware CF (<https://cfw.ftcommunity.de/ftcommunity-TXT/en/programming/python/tutorial-1.html>). Esta guía indica que **para cualquier aplicación es necesario generar 3 archivos:**

- **Manifest:** Este archivo debe contener la descripción de la aplicación a usar, integrando información como nombre, archivo a ejecutar, ícono, etc. (ver ejemplo)

```
[app]
name: BasicControl
category: Tests
author: Name|
icon: icon.png
desc: Control interface for the band
url: http://cfw.ftcommunity.de/ftcommunity-
TXT/en/programming/python/tutorial-1.html
exec: BasicControl.py
managed: yes
uuid: 191fe5a6-313b-4083-af65-d1ad7fd6d269
version: 1.0
firmware: 0.9
```

Nota: El UUID (Código de identificación único) se puede generar el desde este vínculo (<https://www.famkruithof.net/uuid/uuidgen>). Es muy importante que sea **diferente para cada aplicación**.

- **Icon:** El ícono debe adjuntarse en formato JPG o PNG y tendrá un tamaño de 64x64 pixeles.
- **Application:** La aplicación deberá estar construida sobre Python y debe tener una interfaz gráfica básica para poder ejecutarla.

➤ Implementación de las aplicaciones

Para realizar la implementación de las aplicaciones se deben tener en cuenta aspectos claves como el uso de librerías, las definiciones de funciones y clases a trabajar y el control a establecer.

1. Librerías

Para comenzar las implementaciones es indispensable integrar las librerías necesarias para operar las bandas desde el controlador, para ello se debe integrar la ruta de las librerías al archivo y se utiliza el script con las siguientes líneas.

```
#!/usr/bin/env python3
# # -*- coding: utf-8 -*-
```

Nota: Es clave comenzar el script con las líneas mencionadas

Luego se deben importar las librerías a usar. Las más importantes serán ***Touchstyle*** para el control gráfico, ***Ftrobopy*** para el control general y ***smbus*** para el control de los motores (protocolo I2C).

```
import sys
import os
from TouchStyle import *
import ftrobopy
import smbus
```

Nota: No es necesario tener estas librerías en el computador, ya que se ejecutan directamente desde el TXT Controller.

2. Definiciones

Es importante definir **bus** como se muestra a continuación, ya que sobre este se escribe la salida del protocolo I2C para el control de los motores.

```
bus = smbus.SMBus(1) # 1 indicates /dev/i2c-1
```

También se debe definir la **interfaz del programa**. Para esto se define la clase **TouchGuiApplication** que se muestra a continuación. Dentro de esta clase se definen los elementos de la interfaz y las funciones que se usarán en el programa.

```
class TouchGuiApplication(TouchApplication):  
    def __init__(self, args):  
        TouchApplication.__init__(self, args)
```

Dentro de la función **init** se debe incluir la siguiente sentencia, que permite inicializar la ventana y mostrarla en el TXT Controller.

```
self.vbox.addStretch()  
self.win.centralWidget.setLayout(self.vbox)  
self.win.show()  
self.exec_()
```

La ventana se define con la función **TouchWindow** que recibe como parámetro el nombre de la aplicación. Luego se configura el atributo **vbox** como se muestra en la imagen a continuación

```
self.win = TouchWindow("Band Controller")  
self.vbox = QVBoxLayout()  
self.vbox.addStretch()
```

Finalmente, se pueden agregar objetos en la interfaz como se muestra en la imagen inferior. Es importante definir la función del botón que se agrega y que esta función esté definida dentro de la clase **TouchGuiApplication**.

```
buttonP = QPushButton("Prender bus")  
buttonP.clicked.connect(self.turn_on)  
self.vbox.addWidget(buttonP)
```

3. Control

Para controlar las bandas se usa la función *write_byte* de la librería *bus*. Esta función recibe sus parámetros como valores **hexadecimales** y se le debe indicar el puerto de salida y el valor del mismo (velocidad). Los valores se definen para que coincidan con la configuración del Arduino usado para el protocolo I2C, por lo que el puerto de salida siempre será el puerto 8 (Esto según el txt) y los valores para la velocidad estarán entre 0 y 10.

```
bus.write_byte(hex(8), hex(0))
```

4. SSH

La ejecución del código se puede realizar por medio del protocolo **SSH (Secure Shell)**, el cual se puede conectar por medio de una aplicación como **PuTTY**.

Nota: Es importante destacar que esta sección es completamente opcional y no es necesaria; sin embargo, si es **necesario en caso** tal que se desee realizar un **debug** sobre la aplicación creada o revisar algún **output** desde el computador que controla la banda.

A continuación, se muestra el procedimiento para la conexión.

1. Ingresar la IP del controlador en PuTTY con el puerto por defecto y especificando **SSH en el tipo de la conexión**. Luego presionar *Open*.

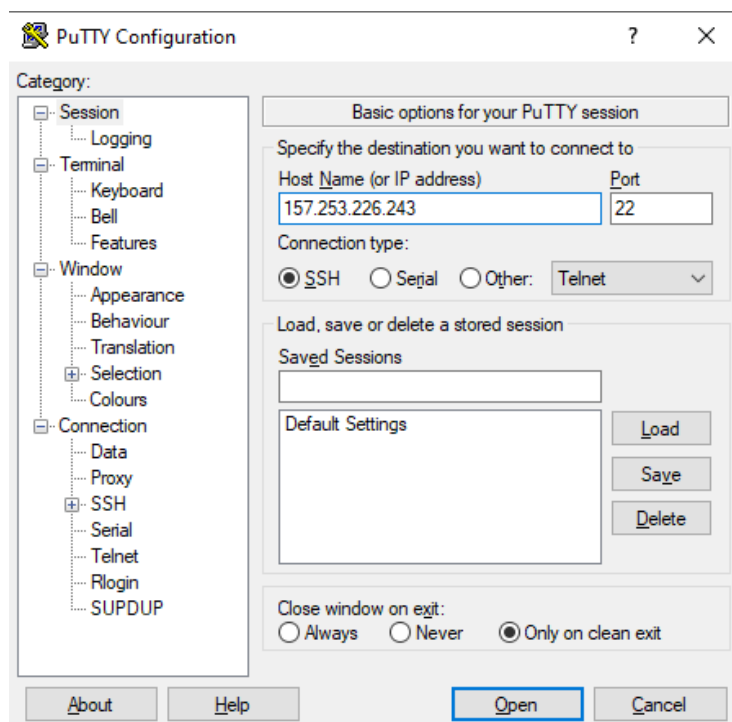
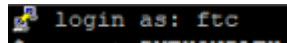
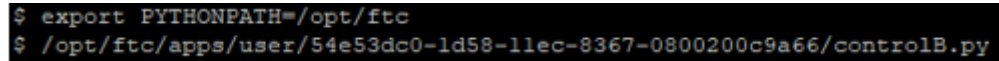


Ilustración 7 Interfaz PuTTY

2. Una vez se abra la ventana correspondiente de manera automática, se debe ingresar el **usuario para login**. El usuario de las bandas es **ftc** y no tiene contraseña.

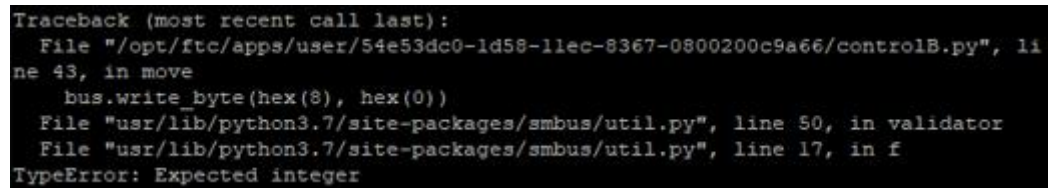
A terminal window showing the prompt "login as: ftc".

3. Se debe ingresar el comando [***export PYTHONPATH=/opt/ftc***] y luego se debe indicar la ruta del archivo a ejecutar, la cual estará dentro de la carpeta *apps* con el UUID (Código de identificación única) que se asignó previamente.

A terminal window showing the commands: `$ export PYTHONPATH=/opt/ftc` and `$ /opt/ftc/apps/user/54e53dc0-1d58-11ec-8367-0800200c9a66/controlB.py`.

Nota: Esta carpeta también debe contener el archivo de Python a ejecutar.

En caso de encontrar algún error, la consola mostrará en dónde se encuentra y por qué se dio:

A terminal window showing a Python traceback error. The text is:

```
Traceback (most recent call last):  
  File "/opt/ftc/apps/user/54e53dc0-1d58-11ec-8367-0800200c9a66/controlB.py", line 43, in move  
    bus.write_byte(hex(8), hex(0))  
  File "/usr/lib/python3.7/site-packages/smbus/util.py", line 50, in validator  
  File "/usr/lib/python3.7/site-packages/smbus/util.py", line 17, in f  
TypeError: Expected integer
```

4. Finalmente, se puede utilizar el comando ***nano*** para editar el archivo de Python desde la consola, guardándose automáticamente en el TXT Controller sin necesidad de volverlo a cargar por la aplicación.

4. Paramiko

Una alternativa para usar SSH sin necesidad de una aplicación adicional es por medio de Python usando la **librería Paramiko**. El funcionamiento de esta librería es muy sencillo y se puede usar de la siguiente forma:

1. Se crea un cliente SSH y se le asigna una autoverificación de *lost keys*, para indicar que se confía en la conexión.

```
ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
```

2. Crear la conexión al TXT con la IP, el usuario *ftc* y la contraseña vacía

```
ssh.connect(hostname='157.253.225.143', username='ftc', password='', port=22)
```

3. Crear una conexión al *shell* (Consola base) y enviar los comandos del SSH por medio de esta. Los comandos por enviar son los mismos que se envían por la aplicación de PuTTY explicada con anterioridad.

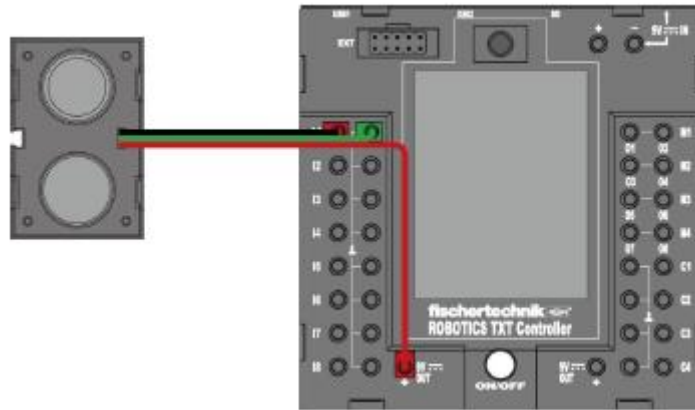
```
shell = ssh.invoke_shell()
shell.send('export PYTHONPATH=/opt/ftc\n')
shell.send('/opt/ftc/apps/user/191fe5a6-313b-4083-af65-d1ad7fd6d210/ultrasound.py\n')
```

Al realizar estos pasos se logra la misma conexión que por medio de PuTTY. Para finalizar siguiendo el procedimiento explicado en el *apartado 2* de este documento correspondiente a la implementación de aplicaciones puede continuar con el proceso.

5. Sensores y actuadores

Para conectar sensores o actuadores de Fischertechnik se usará el datasheet del elemento, el cual puede ser consultado normalmente en internet. Este documento permite saber cómo se conecta el sensor y cómo se realiza la medición. A continuación, se mostrará un ejemplo con un sensor de ultrasonido.

1. Revisar cómo se conecta correctamente el sensor al controlador



2. Configurar la lectura de *ftrobopy*

```
self.txt = ftrobopy.ftrobopy("localhost", 65000)
```

Nota: 65000 corresponde al valor de un puerto único, puede seleccionar el que requiera

3. Usar la función de *ftrobopy* predefinida llamada *ultrasonic* especificando el puerto al que está conectado el sensor.

```
self.ultrasound = self.txt.ultrasonic(1)
```

4. Para revisar la distancia medida se puede llamar al método *distance* de la instancia del ultrasonido creada previamente.

```
self.distance = self.ultrasound.distance()  
print(self.distance)
```

La librería *ftrobopy* tiene más funciones previas que se pueden consultar para diferentes sensores como se encuentra en el archivo adjunto en la carpeta llamado “manual_ftrobopy.pdf.”

6. Troubleshooting

Nota: El problema más recurrente identificado para el manejo del controlador es con la especificación de la ruta.

En ocasiones se tienen problemas con la ejecución de los archivos por el formato de escritura del editor y la interpretación del controlador. Para revisar que esto no ocurra será necesario usar *nano* en *Putty*, desde la carpeta donde se encuentra el archivo de Python en el TXT y cambiar las primeras líneas del código (donde se indica la ruta de las librerías). Esto se hace de la siguiente forma:

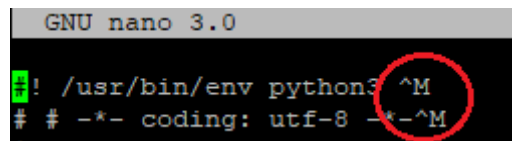
1. Cambiar la carpeta a la ubicación del archivo de Python usando el comando *cd* e ingresando la ruta */opt/ftc/apps/user/* y el código *UUID* (Código de identificación unico) del archivo

```
$ cd /opt/ftc/apps/user/191fe5a6-313b-4083-af65-dlad7fd6d269
```

2. Ingresar el comando *nano* junto con el nombre del archivo *.py*

```
$ nano basicControl.py
```

3. Quitar el *"^M"* de la primera línea. Esto retira la codificación del salto de línea de este comentario, que indica la ubicación de archivos de Python para la ejecución de cualquier código. No es necesario quitarlo para las demás líneas, ya que el inconveniente es únicamente con la especificación de la ruta.



```
GNU nano 3.0
! /usr/bin/env python3 ^M
# -*- coding: utf-8 -*- ^M
```

4. Guardar y salir de la edición

❖ Documentación varia

- Ejemplo de código de uso de las bandas: *BasicControl.py*
- Community Firmware: <https://cfw.ftcommunity.de/ftcommunity-TXT/en/>
- Documentación smbus/
<https://buildmedia.readthedocs.org/media/pdf/smbus2/latest/smbus2.pdf> smbus2:
- Buildroot: <https://buildroot.org/>
- DatasheetTXT Controller:
https://www.google.com/url?sa=i&url=https%3A%2F%2Fcontent.ugfischer.com%2Fcbfiles%2Ffischer%2FZulassungen%2Fft%2FTXT-Controller_es.pdf&psig=AOvVaw1KMn89D92WAE4pn-Ab5jjT&ust=1634164086411000&source=images&cd=vfe&ved=2ahUKewiR5KLg9cXzAhVMZ98KHRv2CrEQr4kDegQIARA3