Plan for documentation

1. Introduction
2. Describe the environment
   1. Hardware
   2. OS and configurations.
      1. Flash Raspbian stretch
      2. Configure Raspbian (VNC, Enable SPI, I2C , X11, etc.)
3. Describe the Project
   1. Goals
   2. VS Project
   3. Premises
4. TO DO list

Index

[Introduction 4](#_Toc522784733)

[Hardware Architecture. 5](#_Toc522784734)

[Operating System 6](#_Toc522784735)

[How to flash Raspbian in T3. 6](#_Toc522784736)

[Configurations required 7](#_Toc522784737)

[Change default password. 7](#_Toc522784738)

[Fix your ethernet and WLAN settings 8](#_Toc522784739)

[Update and Upgrade 9](#_Toc522784740)

[Setup TELNET and enable SSH 9](#_Toc522784741)

[Enabling SPI 11](#_Toc522784742)

[Enabling I2C 12](#_Toc522784743)

[Setup Wiring-Pi 12](#_Toc522784744)

[Test wiringPi’s installation 13](#_Toc522784745)

[Setup mono and enable X11 (X-Server) 14](#_Toc522784746)

[Optional Configurations 17](#_Toc522784747)

[The LED’s Project (Drivers) 17](#_Toc522784748)

[Description 17](#_Toc522784749)

[Original Drivers 18](#_Toc522784750)

[Goals 18](#_Toc522784751)

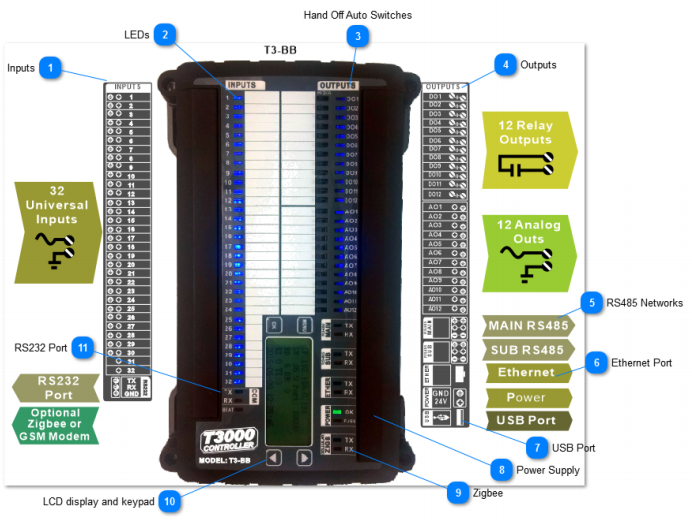
[The .NET Solution. 18](#_Toc522784752)

[Assumptions and project path. 20](#_Toc522784753)

# Introduction

The T3000 controller is a multi-user, including models T3-BB, T3-LB and T3-TB, which can stand alone DDC panel with full communications capabilities. They can be used either stand alone, or in a multiple network system. Multiple communication ports allow the controller to simultaneously operate on a network, host sub networks, and to communicate to local and remote operators.

The T3-BB (Big Brother) has 32 universal inputs, 12 digital outputs and 12 analog outputs.



This document, includes information for a Project specific for T3-BB in order to redesign the communications drivers to access and control Switches and Leds in the top board of the BAC controller. Those drivers will become part of a bigger system, the new generation of T3000 Management Software, the T3000 Cross platform.

T3000 Cross platforms Project, aims for multi-platform enabled software management, capable of running under Windows , Linux distros and Android, where main processor of T3 will be replaced with a Raspberry Pi Compute Module. Programming language is C# with support of .NET Framework 4.0 and Mono.

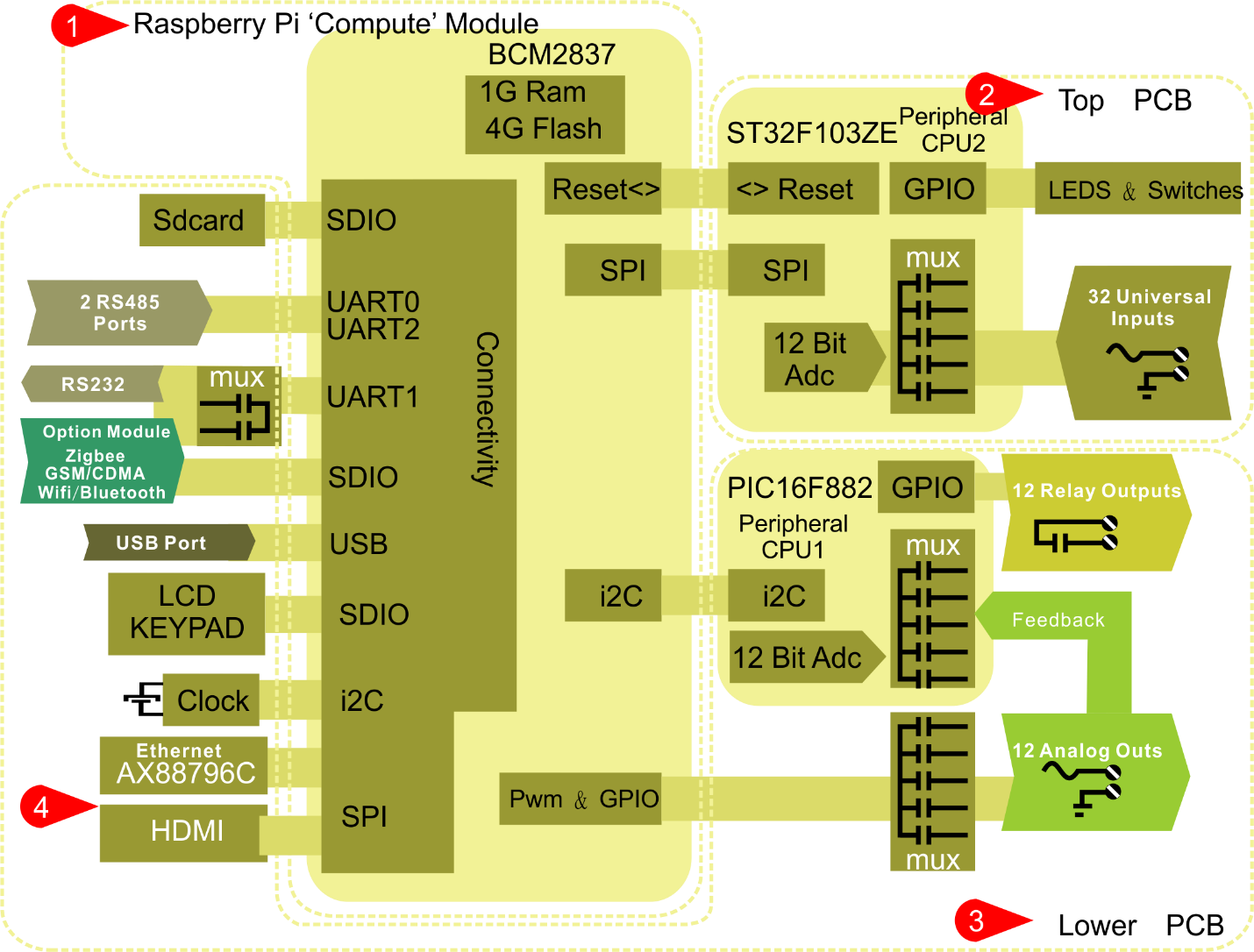
# Hardware Architecture.

The heart of the T3 controllers is a 32 bit, ST32F103ZE Arm processor from the M3 core family. This is a high performance CPU operating at up to 120 Mips which is plenty of processing power for controlling typical HVAC applications while maintaining fast and responsive communications.

Earlier versions of the product shipped before Q1 2017 use the Asix AX11015 which is a general purpose CPU running at 100 Mips with an on board Ethernet controller. Due to some issues handling multiple connections over TCP we migrated to the Arm based chip which has solved that as well as opened up other capabilities thanks to the extra serial ports on the Arm chip.

The 200 pin SODIMM socket which hosts the main CPU can also be populated with a 'Raspberry Pi compute' module which runs the Rasbian operating system, and now also the Win10 IoT environment. There is currently not much of software support for the Pi version but the hardware is in production and the open source project behind it is slowly gaining steam**: T3000 Cross platform project.**

**In this project, wherever mentioned the main CPU will be a Raspberry Pi compute module v1, or CM1.**

Figure 3: Lower PCB to the left, Top PCB to the right.

The hardware is made up of a top PCB and lower PCB. The Raspberry Pi compute module is plugged into a socket on the lower PCB, showing at Tab1 in the image above. (CM1 is a BCM2835 ARMv6 at 700 MIPS, not a BCM2837)

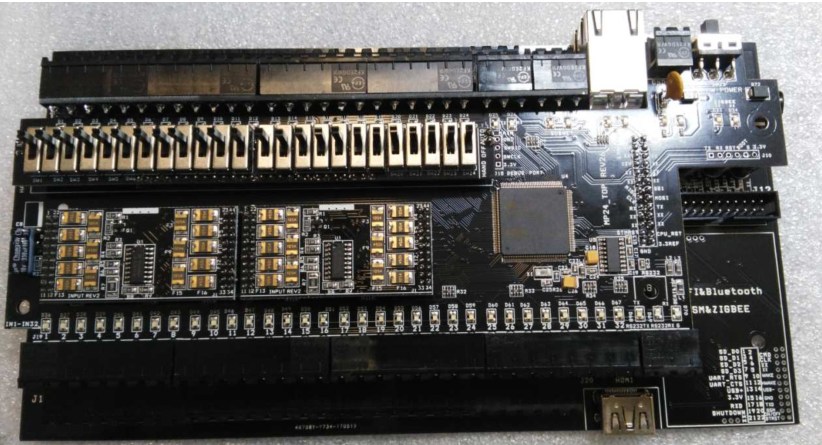


Figure 4: Top PBC with STM CPU, switches and Leds

The GPIO of the Pi are used by the LCD, UARTS and SD card so expansion to the rest of the hardware is offloaded to two auxiliary CPUs. On the top board at Tab2 is the ST32F103ZE which is responsible for managing the LEDS, hand-off-auto switches and the universal inputs. On the bottom board at Tab3 is the second CPU, a PIC16F882 which is responsible for managing the relays and analog outputs. The Pi communicates to the two auxiliary CPU’s using SPI commands.

The CM1 contains a BCM2835 processor (as used on the original Raspberry Pi and Raspberry Pi B+ models), 512MByte LPDDR2 RAM and 4Gbytes eMMC Flash. The pinout of CM1 and CM3 are identical.

# Operating System

Raspbian GNU/Linux 9 (stretch) OS need to be installed in CM1, as original OS is a little bit outdated (Raspbian Jessie or lower)

Raspbian Stretch is a Linux raspberrypi 4.9.59+ #1047 Sun Oct 29 11:47:10 GMT 2017 armv6l GNU/Linux, special distribution based on Debian.

## How to flash Raspbian in T3.

Flasing new OS into CM1 of T3 (tipically Raspbian), requires removing (unscrewing) the top PBC, to access J15 jumper in lower PCB.

1. Put J15 in USB mode.
2. Connect T3 to PC using USB, Power ON
3. Install drivers and boot tool
4. Once recognized new hardware as BCM27xboot. Run RPI Boot to use the BCM27XBOOT as new drive.
5. Use Win32DiskImager to flash new Raspbian image into the drive.
6. When flashing is done, Just Power OFF, disconnect and put J15 in FLASH mode.
7. Power On. Setup of new OS will initiate. (You will need a USB keyboard and HDMI screen or TV connected to T3 before powering ON.)

|  |  |
| --- | --- |
| **Rasbian Images** | https://www.raspberrypi.org/downloads/raspbian/ |
| **Rpi boot Software** | https://www.raspberrypi.org/documentation/hardware/computemodule/cm-emmc-flashing.md |
| **Drivers for BM** | https://www.raspberrypi.org/documentation/hardware/computemodule/cm-emmc-flashing.md |
| **Win32Imager** | https://sourceforge.net/projects/win32diskimager/?source=typ\_redirect |

## Configurations required

This is a basic guide to configuration and setup required to work with this project (on a T3-BB + CM1, running Rasbian Stretch). Some knowledge on linux commands is advisable.

T3000 Cross platform and drivers must be runnable in Raspbian. So, we are going to need mono support for executables (.Net 4.0). Additional configuration required for enabling GPIO as SPI using raspi-config utility and for remote connection to X-Windows via SSH and VNC.

### Change default password.

User management in Raspbian is done on the command line. The default user is pi, and the password is raspberry. You can add users and change each user's password.

Once you're logged in as the pi user, it is highly advisable to use the passwd command to change the default password to improve your Pi's security.

Enter passwd on the command line and press Enter. You'll be prompted to enter your current password to authenticate, and then asked for a new password. Press Enter on completion and you'll be asked to confirm it. Note that no characters will be displayed while entering your password. Once you've correctly confirmed your password, you'll be shown a success message (passwd: password updated successfully), and the new password will apply immediately.

**The default pi user on Raspbian is a sudoer.** This gives the ability to run commands as root when preceded by sudo, and to switch to the root user with sudo su.

To add a new user to sudoers, type sudo visudo (from a sudoer user) and find the line root ALL=(ALL:ALL) ALL, under the commented header # User privilege specification. Copy this line and switch from root to the username. To allow passwordless root access, change to NOPASSWD: ALL. The example below gives the user bob passwordless sudo access:

# User privilege specification

root ALL=(ALL:ALL) ALL

bob ALL = NOPASSWD: ALL

Save and exit to apply the changes. Be careful, as it's possible to remove your own sudo rights by accident.

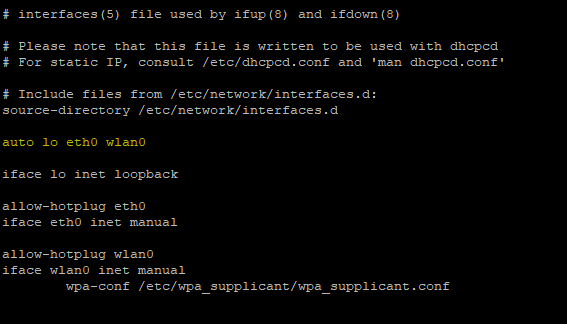
**You will need a non-blank password for remote access to terminal using SSH, don’t leave it empty!**

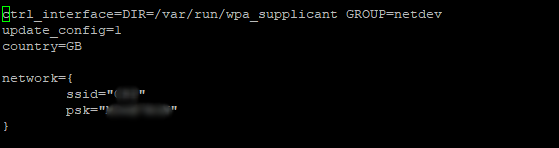
### Fix your ethernet and WLAN settings

You will need to update Raspbian via internet, and later on for remote access to T3 at least one or both of the network interfaces. T3-BB-321 only has one integrated ethernet adapter (RJ45 port). When you install Raspbian, this configuration is not ready for production.

Edit and reconfigure /etc/network/interfaces and /etc/wpa\_supplicant/wpa\_supplicant.conf, in order to allow WIFI connection.

ie: ETH0 and WLAN with auto config (DHCP Enabled)



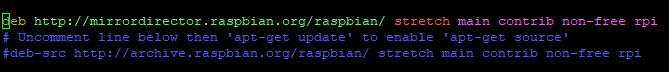


Write your SSID and Password in /etc/wpa\_supplicant/wpa\_supplicant.conf for wireless access. Note, that you can use a wireless nano adapter attached to USB Port of T3, but only when you no longer need a keyboard attached, namely when you are ready to use remote access. While so, other option is, using a USB 3.0 Hub to connect KB and Nano WiFI). We have tested a TP-LINK nano wifi USB adapter, and it doesn’t required adittional or third party drivers.

Reboot, and test your network and internet access.

### Update and Upgrade

Before using typical commands: apt-get update & apt-get upgrade, you have to edit and fix your source repositories file in /etc/apt/sources.list to include at least the one listed as example.



Update and upgrade with this commands.

$ sudo apt-get update

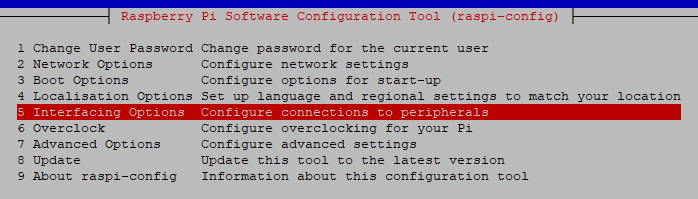
$ sudo apt-get upgrade

### Setup TELNET and enable SSH

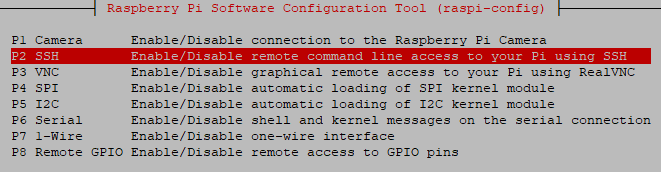
sudo apt-get install telnetd

sudo raspi-config

Select item “5. interfacing options”, and don’t touch anything on top of this option.



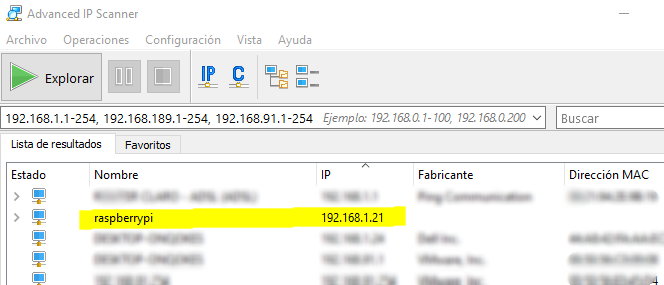
Enable option P2. SSH. It will install SSH package.



Later on, we are going to use raspi-config to setup and enable automatic loading of SPI and I2C.

So, now it’s time to test your network and remote access using putty. If you did auto configuration for network interfaces, then you’ll have to investigate the ip assigned by DHCP, you could use a tool like Advanced IP Scanner from windows to do so, or open up a terminal console in Raspbian and type in:

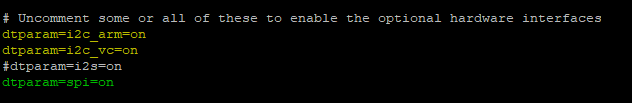
hostname -I



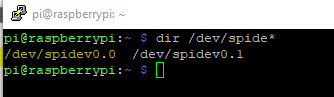
### Enabling SPI

The Raspberry Pi is equipped with one SPI bus that has 2 chip selects.

The SPI master driver is disabled by default on Raspbian. To enable it, use raspi-config, or ensure the line dtparam=spi=on isn't commented out in /boot/config.txt, and reboot.



If the SPI driver was loaded, you should see the device /dev/spidev0.0.



The SPI bus is available on the P1 Header:

MOSI P1-19

MISO P1-21

SCLK P1-23 P1-24 CE0

GND P1-25 P1-26 CE1

There are two preferred ways to easy access to Pi SPI.

1. **WIRINGPI**

WiringPi is a PIN based GPIO access library written in C for the BCM2835, BCM2836 and BCM2837 SoC devices used in all Raspberry Pi. versions. It’s released under the GNU LGPLv3 license and is usable from C, C++ and RTB (BASIC) as well as many other languages with suitable wrappers, including C#.

It provides access to GPIO and other IO functions on the Broadcom BCM 2835 chip. Accesses the hardware registers directly.

<http://wiringpi.com/>

1. **BCM2835 LIBRARY**

This is a C library for Raspberry Pi (RPi). It provides access to GPIO and other IO functions on the Broadcom BCM 2835 chip. Accesses the hardware registers directly.

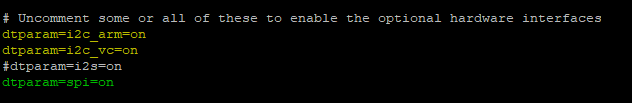
<http://www.airspayce.com/mikem/bcm2835/>

**The BCM2835 on the Raspberry Pi has 3 SPI Controllers. Only the SPI0 controller is available on the header.**

### Enabling I2C

Use raspi-config or uncomment in file /boot/config.txt (yellow lines).

Reboot.



### Setup Wiring-Pi

WiringPi includes a command-line utility **gpio** which can be used to program and setup the GPIO pins. You can use this to read and write the pins and even use it to control them from shell scripts

**To install… First check that wiringPi is not already installed. In a terminal, run:**

$ gpio -v

If you get something, then you have it already installed. The next step is to work out if it’s installed via a standard package or from source. If you installed it from source, then you know what you’re doing – carry on – but if it’s installed as a package, you will need to remove the package first. To do this:

$ sudo apt-get purge wiringpi

$ hash -r

**Then carry on.**

**If you do not have GIT installed, then under any of the Debian releases (e.g. Raspbian), you can install it with:**

$ sudo apt-get install git-core

**If you get any errors here, make sure your Pi is up to date with the latest versions of Raspbian: (this is a good idea to do regularly, anyway)**

$ sudo apt-get update

$ sudo apt-get upgrade

**To obtain WiringPi using GIT:**

$ cd

$ git clone git://git.drogon.net/wiringPi

**If you have already used the clone operation for the first time, then**

$ cd ~/wiringPi

$ git pull origin

**Will fetch an updated version then you can re-run the build script below.**

**To build/install there is a new simplified script:**

$ cd ~/wiringPi

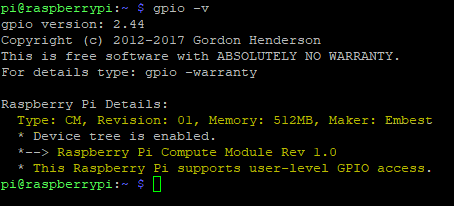
$ ./build

**The new build script will compile and install it all for you – it does use the sudo command at one point, so you may wish to inspect the script before running it.**

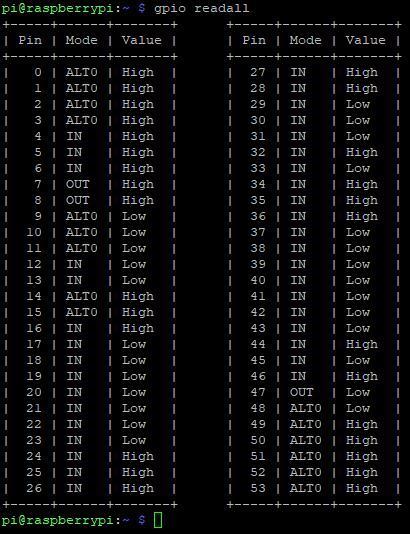
### Test wiringPi’s installation

**run the gpio command to check the installation:**

$ gpio -v



$ gpio readall



That should give you some confidence that it’s working OK.

*WiringPi is released under the GNU Lesser Public License version 3.*

### Setup mono and enable X11 (X-Server)

This is a required step to be able to run .net program in Raspbian stretch. Also, you may want to do a remote connection via VNC to easy testing of the LED’s project.

Note that without a X-Windows server and mono you cannot execute any .net program that uses windows forms.

Assuming you have all system clean and updated. Follow this guide.

If you are not sure about that, then execute these commands:

sudo apt-get update

sudo apt-get upgrade

sudo apt-get dist-upgrade

sudo apt-get clean

#### Installing Mono.

**Clean any leftovers from previous installations.**

sudo apt-get clean

**Install DirMngr as is required for the next step, without it everything will fail.**

sudo apt-get install dirmngr

**Prepare raspbian 9 to download latest versions of mono**

sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 3FA7E0328081BFF6A14DA29AA6A19B38D3D831EF

echo "deb http://download.mono-project.com/repo/debian raspbianstretch main" | sudo tee /etc/apt/sources.list.d/mono-official.list

sudo apt-get update

**Setup latest version of mono**

sudo apt-get mono-complete

**Ensure .net 4.5 compatibilty.**

sudo apt-get install libmono-system-core4.0-cil

#### Setup and Enable X11

In order to have a GUI, we need these 4 things:

1. Display Server
2. Desktop Environment
3. Window Manager
4. Login Manager

Since we need 4 things, to make life easier, these 4 things are:

1. Xorg Display Server
2. Raspberry Pi Desktop (RPD) or Lightweight X11 Desktop Environment (LXDE) or XFCE Desktop Environment (XFCE) or MATE Desktop Environment (MATE)
3. Openbox Window Manager (RPD/LXDE) or XFWM Window Manager (XFCE) or Marco Window Manager (MATE)
4. LightDM Login Manager
5. **Setup xorg**

sudo apt-get install --no-install-recommends xserver-xorg

sudo reboot

Setup xinit, required to start X-Server from console. (optional: but recommended as plan B, when Raspbian is unable to start automatically X-Server after boot up, then you could start X-server form a console.)

sudo apt-get install --no-install-recommends xinit

sudo reboot

1. **Setup RPD (Raspberry Pi Desktop)**

sudo apt-get install raspberrypi-ui-mods

sudo reboot

1. Openbox Window Manager is installed by default when you install RPD. You do not need to do anything here.
2. Finally, we need to install LightDM login manager. If you have installed the basic or stripped version of the RPD desktop environment, then LightDM package is installed automatically so you can skip this step. As of the release of Raspbian Stretch, if you installed the XFCE desktop environment, LightDM is also installed automatically so you do not need to do this step. Otherwise, to install LightDM, type in:

sudo apt-get install lightdm

sudo reboot

Note: If you choose not to install a login manager but you did install xinit earlier, then this means that everytime your Pi boots, you will always boot into the command line. However, the desktop environment is already set up for you and is ready to be launched at any time.

if no login manager was installed, then just login via the command line! At anytime, you can launch the Xorg Display Server by typing in:

startx

### Optional Configurations

You may find usefull to setup a ftp server to upload your files using WinSCP for FTPS mode.

sudo apt-get install ftpd

sudo reboot

And finally, allow remote connection to your system using VNC server. (Requires GUI X-Server previously setup and started).

sudo apt-get install realvnc-vnc-server

sudo reboot

# The LED’s Project (Drivers)

## Description

LED’s project is a rewriting using C# .Net 4.0 of drivers of previous T3000 (MFC) written in C and C++.

Original drivers, based on STM32F10x Standard Peripherals Library, are used to initialize GPIO for STM32F103ZE and allow SPI communication between connected devices in TOP PBC and main CPU.

One important note to keep in mind is that TOP PCB also contains a STM32F103ZE which will respond to same commands as in original version of drivers, but main CPU is replaced by RPI CM1 in this project.

As you are already informed, TOP PCB includes LEDS, inputs and switches.

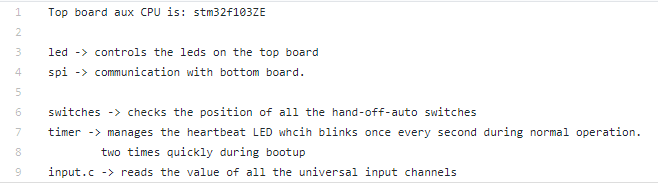
### Original Drivers

Original drivers (a least main files or part of them) are located in same GITHUB repository for T3000 Cross Platform.

<https://github.com/temcocontrols/T3000_CrossPlatform.git>

Under the folders/subfolders: /docs/CodeToBePorted/HardwareDrivers/

TopBoardNotes.txt shows this information for subfolder /TOP



So, when inspecting TOP Folder, you are basically seeing what should be done (rewritten) in this new LED’s projects.

Original drives, of course allow a lot more specifics functionalities, as UART and serial interfaces management, OUTPUTS management, SD management, etc.

Note that main.c references stm32fx10.h (STM32F10x Standard Peripherals Library) which it’s not included in the repository.

## Goals

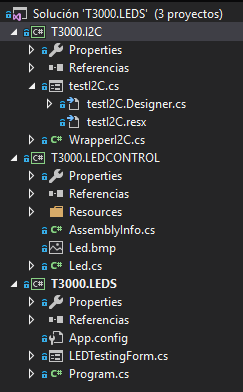
Main goal is to provide a GUI of T3000 written in C# for raspbian, ready to show and change current state of leds in TOP PCB. This GUI is going to be included in T3000 Cross platform.

Also, the project, should be able to provide same interface to manage bidirectionally not only led states, but inputs and switches.

This project (solution) is not intended to extend into a MODBUS stack.

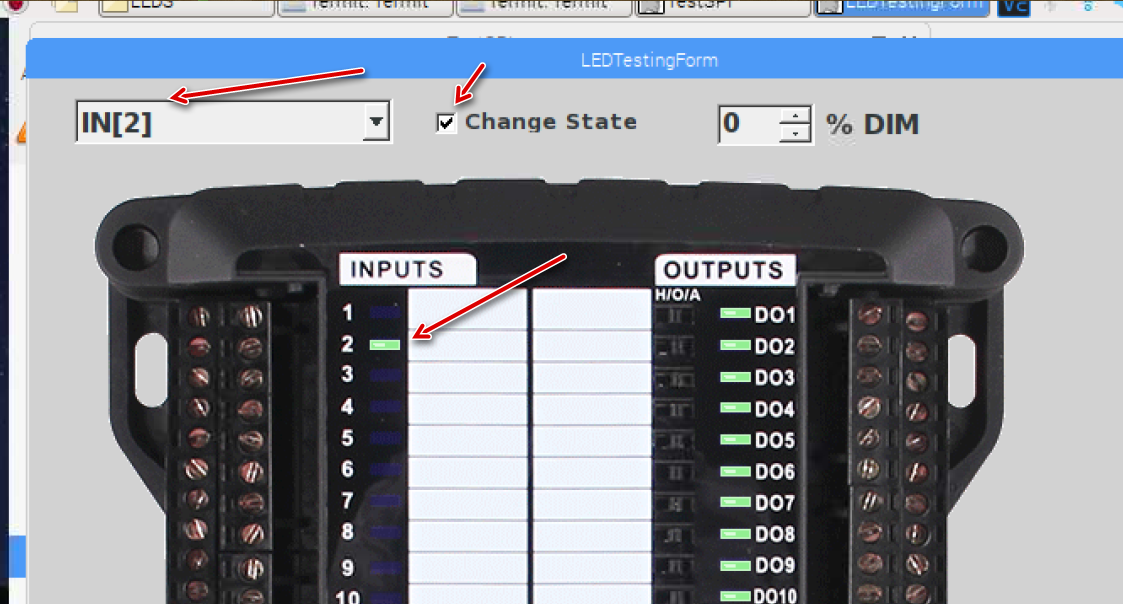
## The .NET Solution.

.Net solution named T3000.Leds is written in C#, and contains 3 projects:

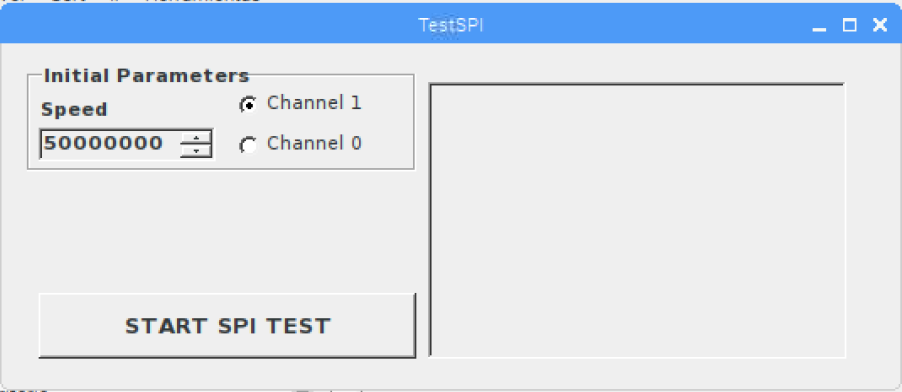
1. T3000.LEDS is an executable .Net for launching the GUI
2. T3000.LEDCONTROL, a Class Library project for LEDs GUI, that defines a Led control dimmable and scalable that is painted over a realistic picture of a T3 (only T3-BB) at this moment. T3000.LEDCONTROL class library can not be used under Windows, as it requires direct connection with GPIO, SPI a I2C interfaces under Raspbian.
3. T3000.I2C project is a Class library in early stages, of testing SPI and I2C communications by means of WiringPi library wrappers for .NET (Wiring Pi is written using C Language)

All T3000.LEDS solution is targeted against .NET 4.5, it must be downgraded to .NET 4.0 when integrated into T3000 Cross Platform.

Also note that T3000.I2C, has wrong name. It has to be renamed to T3000.Dirvers or so.



This is how it looks like the GUI for Leds.



This a primitive test form for SPI. Not even functional right now because this project does not reconfigure GPIO pins.

\*\*\* More testing and demo images to add here.

## Assumptions and project path.

AS is not easy to rewrite STM32FX10 library C code to C# .Net, first assumption is that there is another library ready to use for Pi and .net. And there is, WiringPi DLL and wrappers from wiringpi.com, a GPIO Interface library for the Raspberry Pi.

Chances are, that WiringPi can do the basics or same as STM32FX10 library. Otherwise, we must take the long way and recode STM drivers.

Based on new information, that it’s going to be shared in the “To Do List” section, basics steps to achieve SPI communication between CM1 and TOP PCB using WiringPi would be as follow: