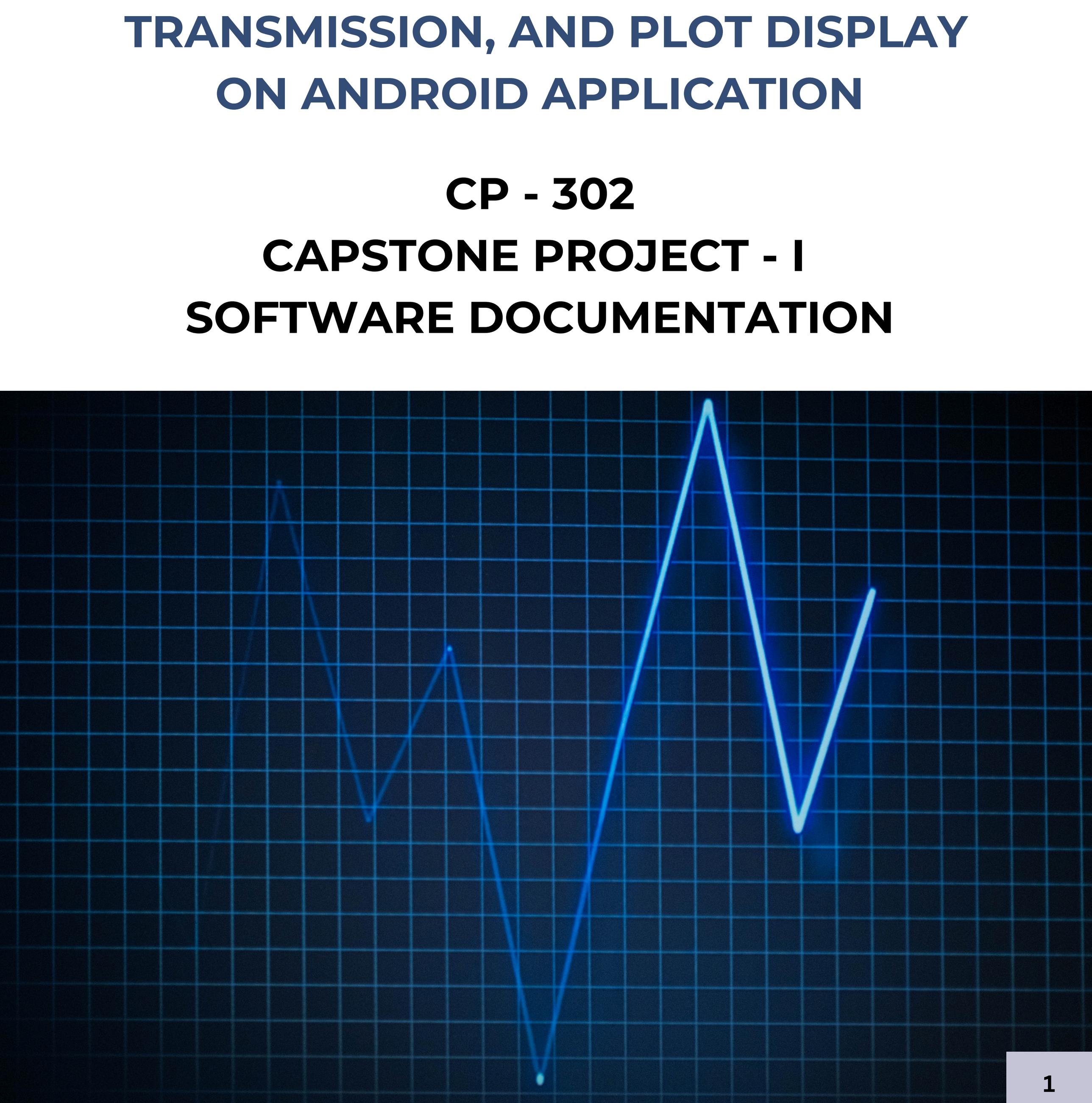
ECG SIGNAL ACQUISITION, TRANSMISSION, AND PLOT DISPLAY ON ANDROID APPLICATION

CP - 302
CAPSTONE PROJECT - I
SOFTWARE DOCUMENTATION



Submitted To : Dr. JS Sahambi

Submitted By :

Aditya Gupta 2022EEB1149

Jalpan Upadhyay 2022EEB1177

Index

Project Overview

Through this project we aim to deliver a lightweight tele-ECG solution that continuously captures multichannel ECG signals, streams them wirelessly to a patient’s smartphone, uplinks the data to the cloud, and makes it instantly available for review by clinicians on desktop or mobile—enabling real-time remote cardiac monitoring and timely intervention.

Work flow of the Project

- **ECG Acquisition**
  - A small analog frontend (AFE) and NXP microcontroller collect raw ECG signals from the patient’s electrodes.
- **Data Injection into WBZ451**
  - The processed ECG samples are fed into the WBZ451 BLE module, which runs FreeRTOS tasks and timers to manage real-time sampling and buffering.
- **BLE Transmission to Smartphone**
  - WBZ451 advertises and, upon connection, pushes ECG packets over Bluetooth Low Energy to the companion Android app.
- **On-Device Visualization**
  - The smartphone app plots each lead live, offers controls (e.g., channel selection, zoom, cursors), and handles basic user interactions like LED toggling or alerts.
- **Cloud Uplink**
  - The app batches and uploads incoming ECG data to a secure cloud database in real time, ensuring encrypted storage and audit trails.
- **Clinician Access**
  - Doctors log into a web or desktop portal to stream or review historical ECG records, annotate events, and generate reports—delivering actionable insights no matter where they are.

Software

MPLAB X IDE really ties everything together—from setting up peripherals to flashing code and debugging right on the WBZ451 board. When we launch our project, the IDE automatically talks to the PICkit On-Board 4 debugger so we can load our firmware, pause execution wherever we need, peek at registers and memory, and watch variables change in real time as our Zigbee or BLE code runs. With the MCC plugin, we simply drag and drop blocks for timers, UART, SPI, GPIO or radio middleware, click to generate clean, commented setup code, and then tweak it right in the editor. Before we even touch the hardware, the built-in simulator and logic-analyzer give us confidence that our timing and signals look right, and the reporting features let us keep an eye on code coverage and binary size—crucial for squeezing a wireless stack into a small microcontroller. In short, MPLAB X IDE makes our firmware workflow smoother and gives us the visibility and control we need to build reliable, low-power IoT designs.

Configuring MPLAB Software

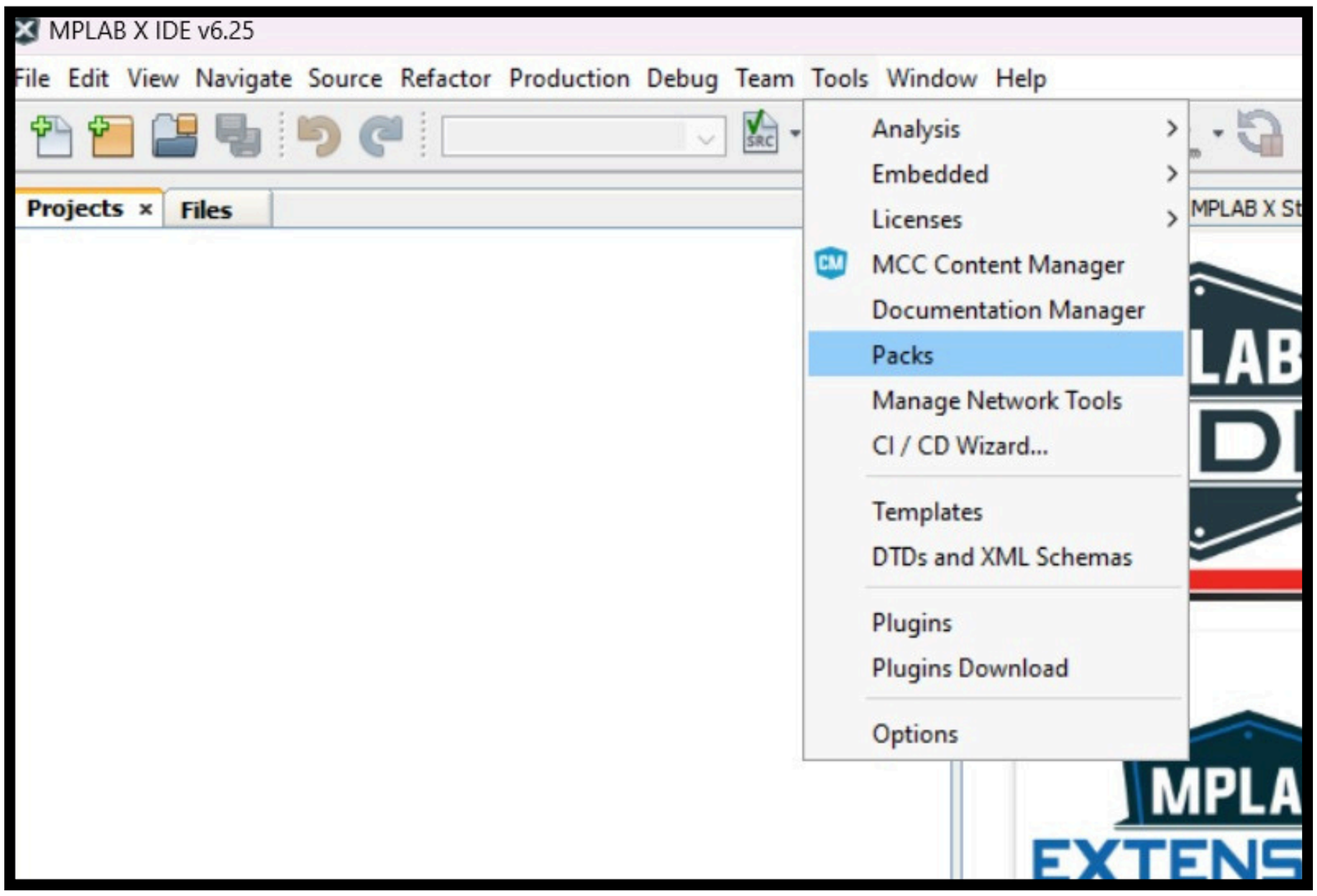
Installing IDE and Compiler:

- Go to https://www.microchip.com/en-us/tools-resources/develop/mplab-x-ide#tabs . And download the latest version of MPLABxIDE (6.25)

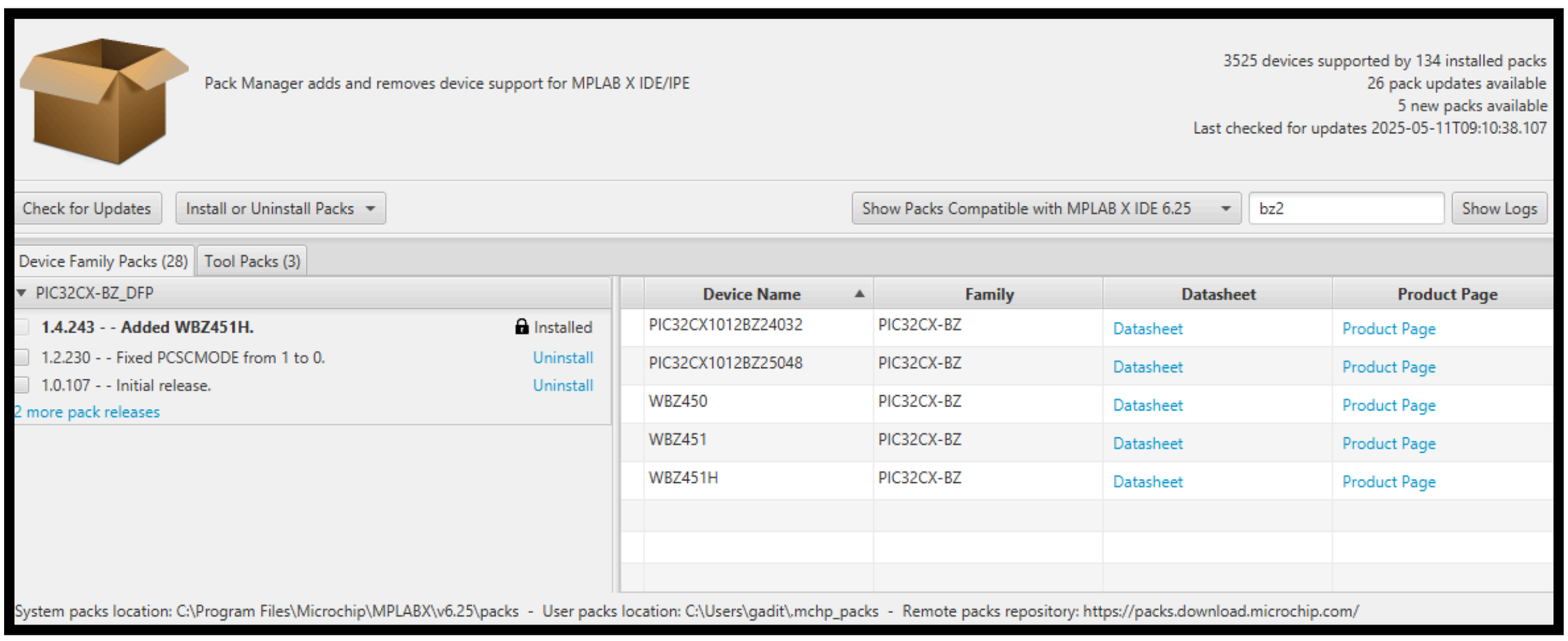
Title	Version Number	Date	
MPLAB X IDE (Windows) bcdd2e14... 6745	6.25	04 Mar 2025	Download
MPLAB X IDE (Linux) ed015d1f... 91ad	6.25	04 Mar 2025	Download
MPLAB X IDE (macOS) 289fd90e... 8292	6.25	04 Mar 2025	Download
MPLAB X IDE Release Notes	6.25	04 Mar 2025	Download

- Download XC32 compiler by going to https://www.microchip.com/en-us/tools-resources/develop/mplab-xc-compilers/xc32 and download the latest version of XC32 compiler (4.6)

Installing Device family part packs:

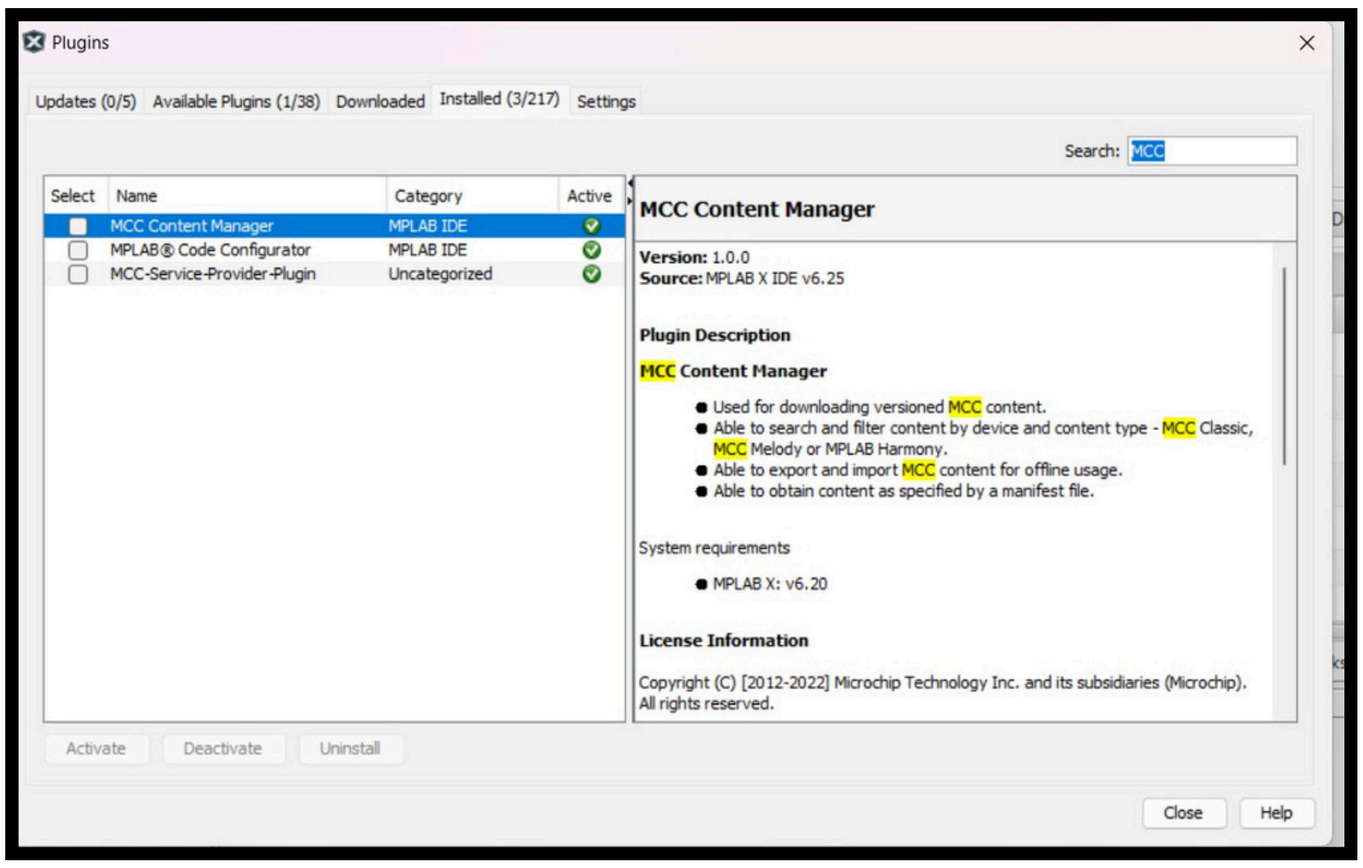


- In the MPLAB X IDE window, select Tools tab and from the drop down menu, select *Packs*.
- Search for "BZ2" in the search box available and select *Install*

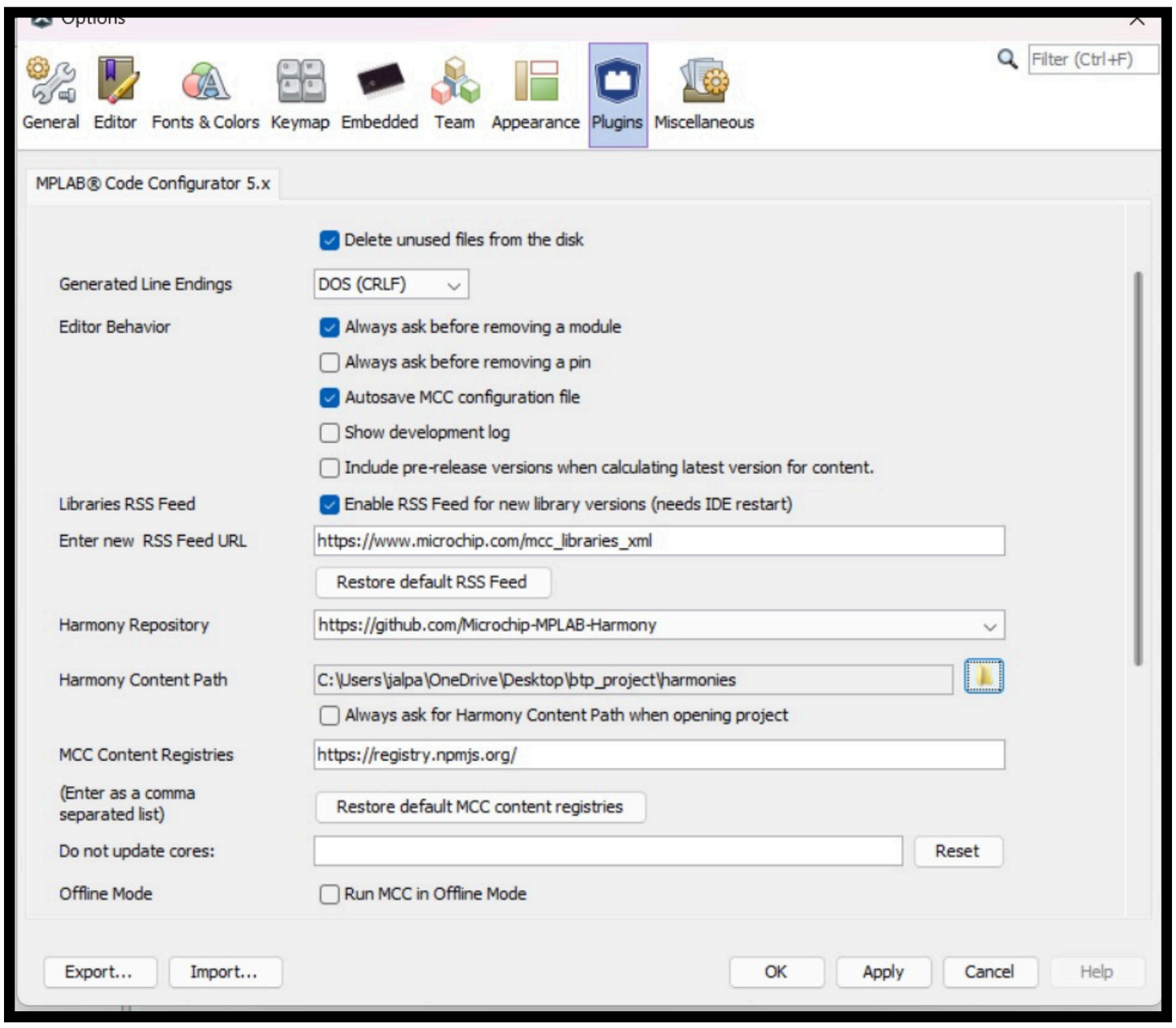


Installing the MCC Plugins:

- In the MPLAB X IDE window, select Tools tab and from the drop down menu, select Plugins.
- Search for "MCC" in the search box available in Available Plugins and Install MPLAB Code Configurator.



- Several aspects of the operation of the MCC can be managed by using the “Options” panel, which can be invoked by clicking Tools → Options → Plugins.
- In the "Harmony Content Path" specify the location where the Harmony 3 repositories should be downloaded with content manager or cloned from GitHub repository. Click Apply and OK

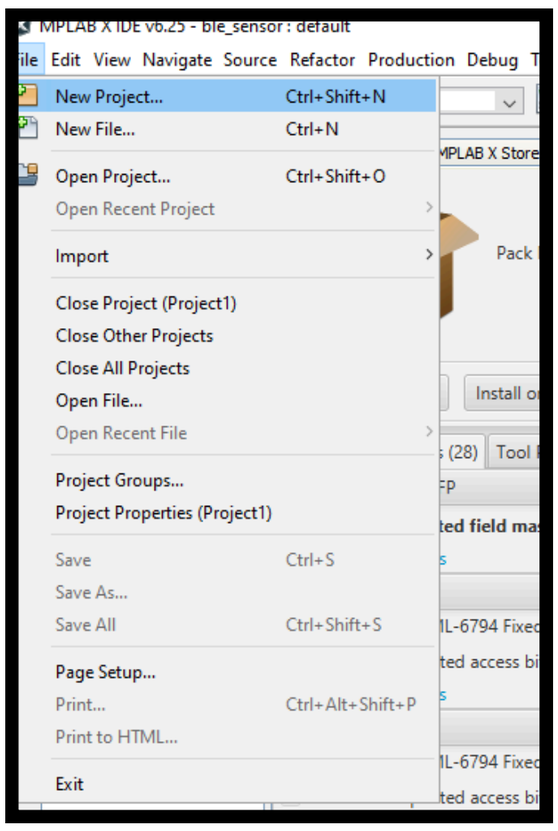


Installing Harmony 3 Dependencies:

Create a new “MCC Harmony” project (Create an empty project and clone the required repositories to clone the Harmony repositories)

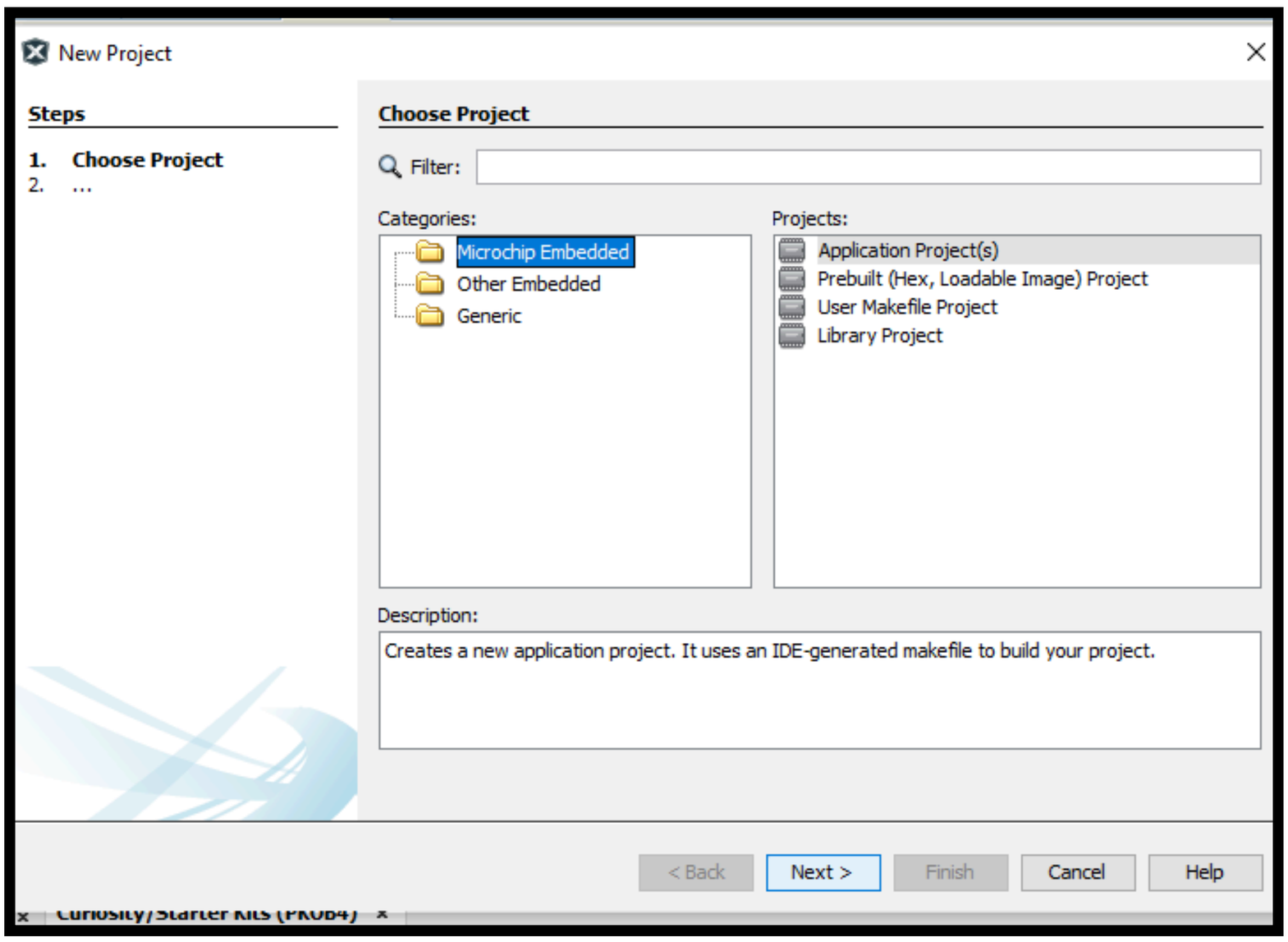
Step 1

In MPLAB X IDE, select File -> New Project



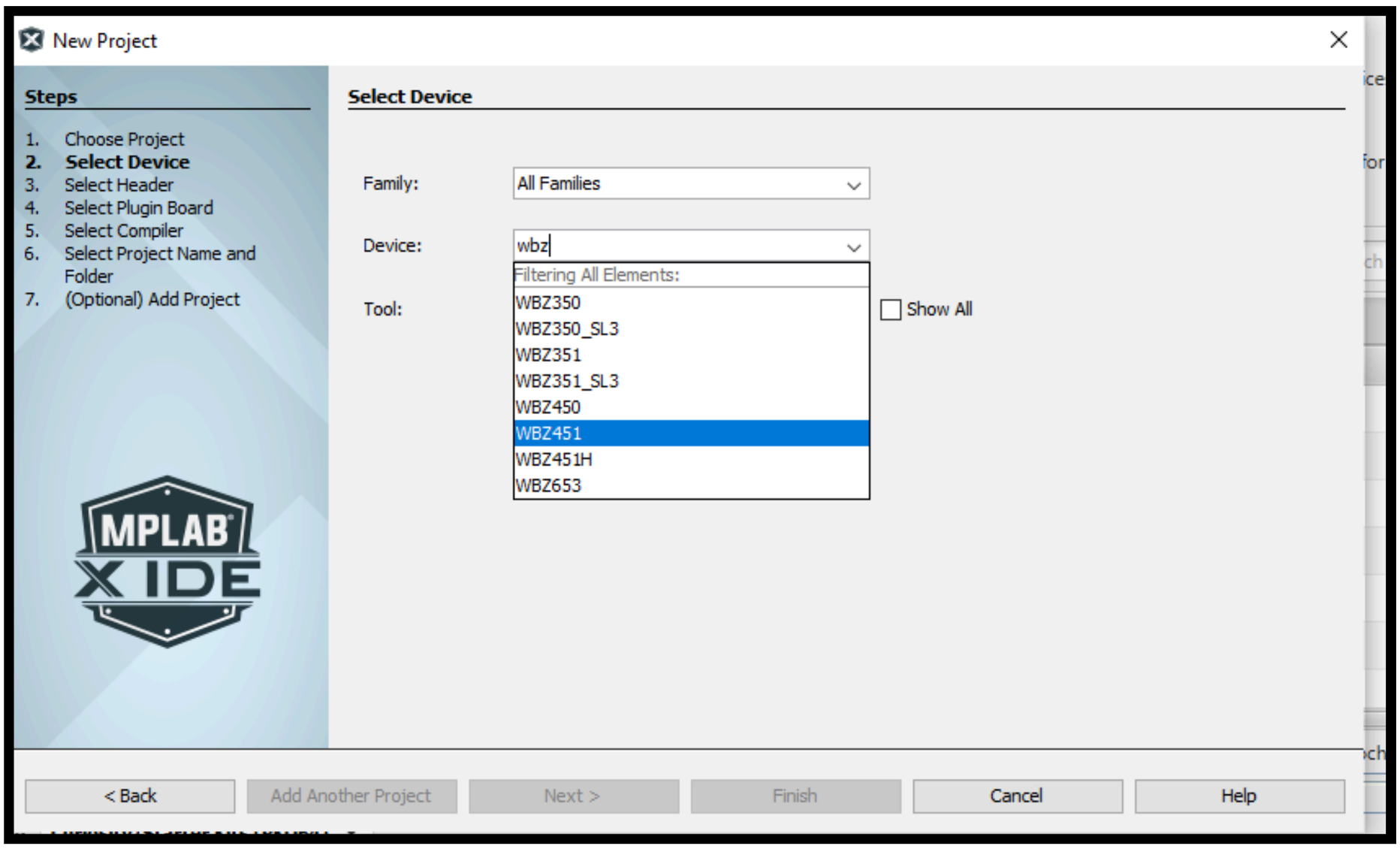
Step 2

Select Microchip Embedded. Click on next.



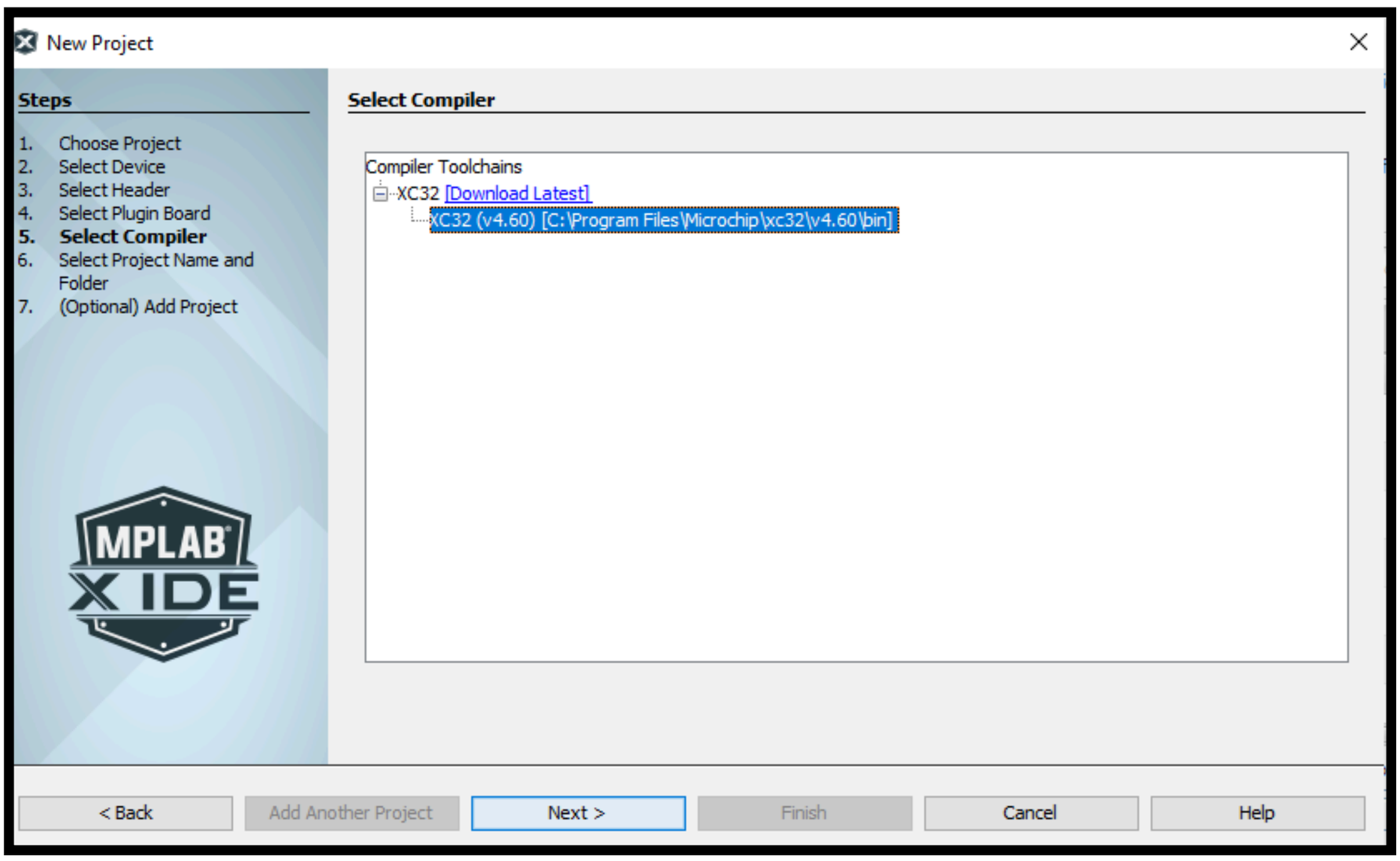
Step 3

In Device drop down menu select WBZ451



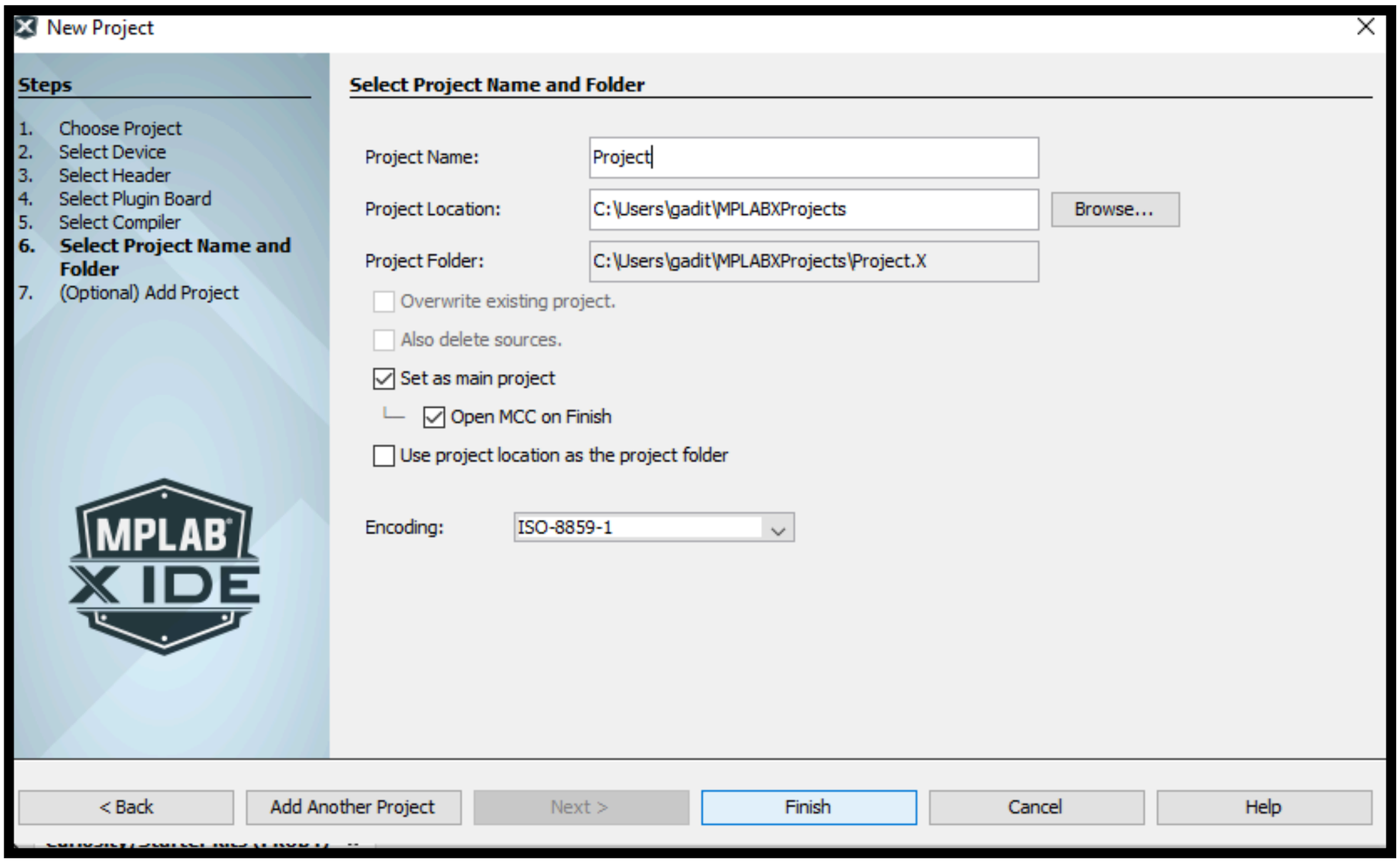
Step 4

Select XC32 as compiler and click on next



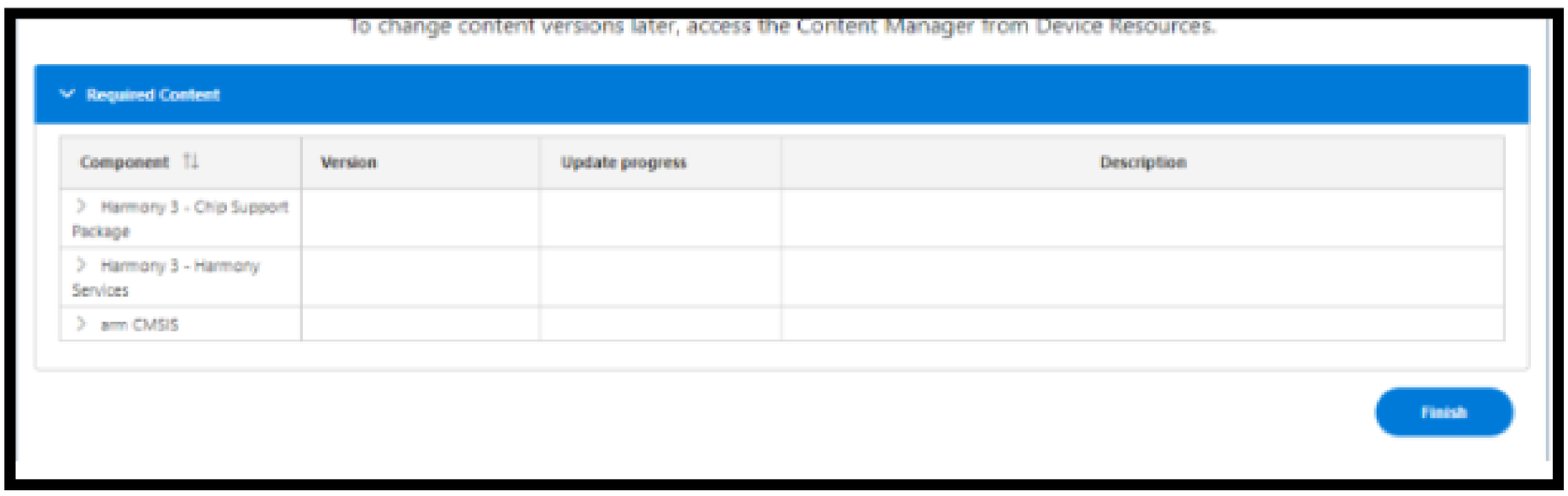
Step 5

Give Project Name and Path, and then click on Finish.



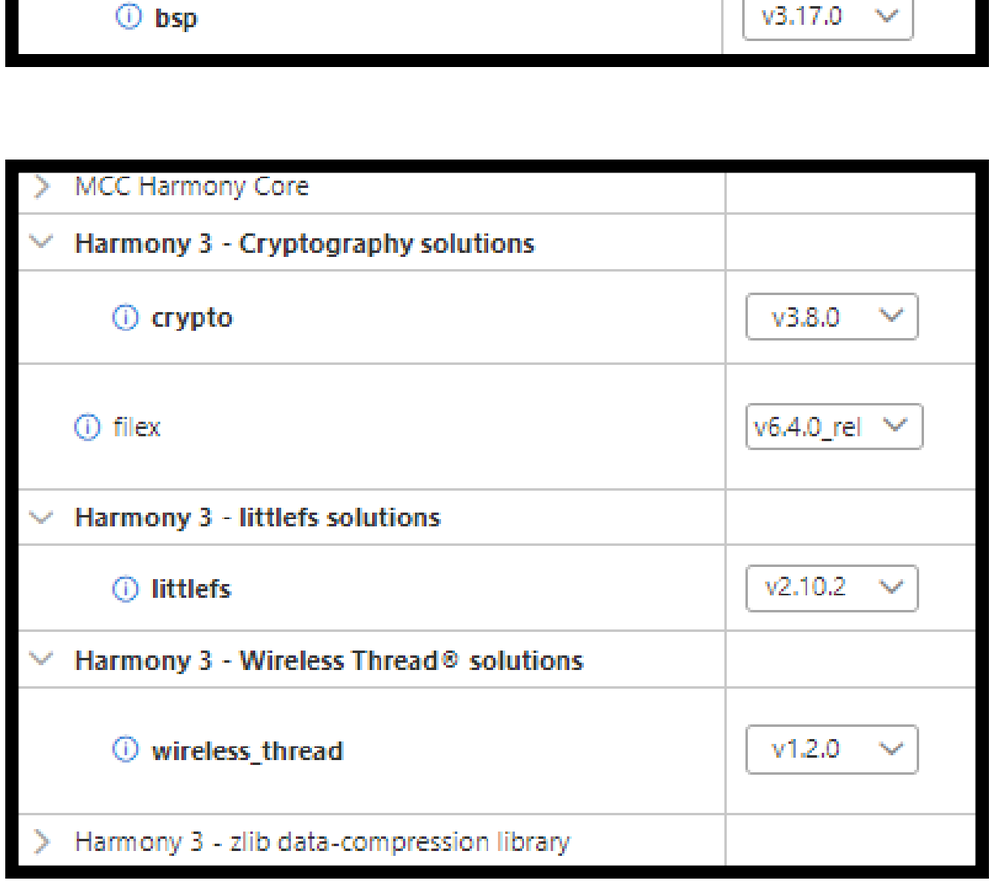
Step 6

Click on finish, to install the required harmony content.



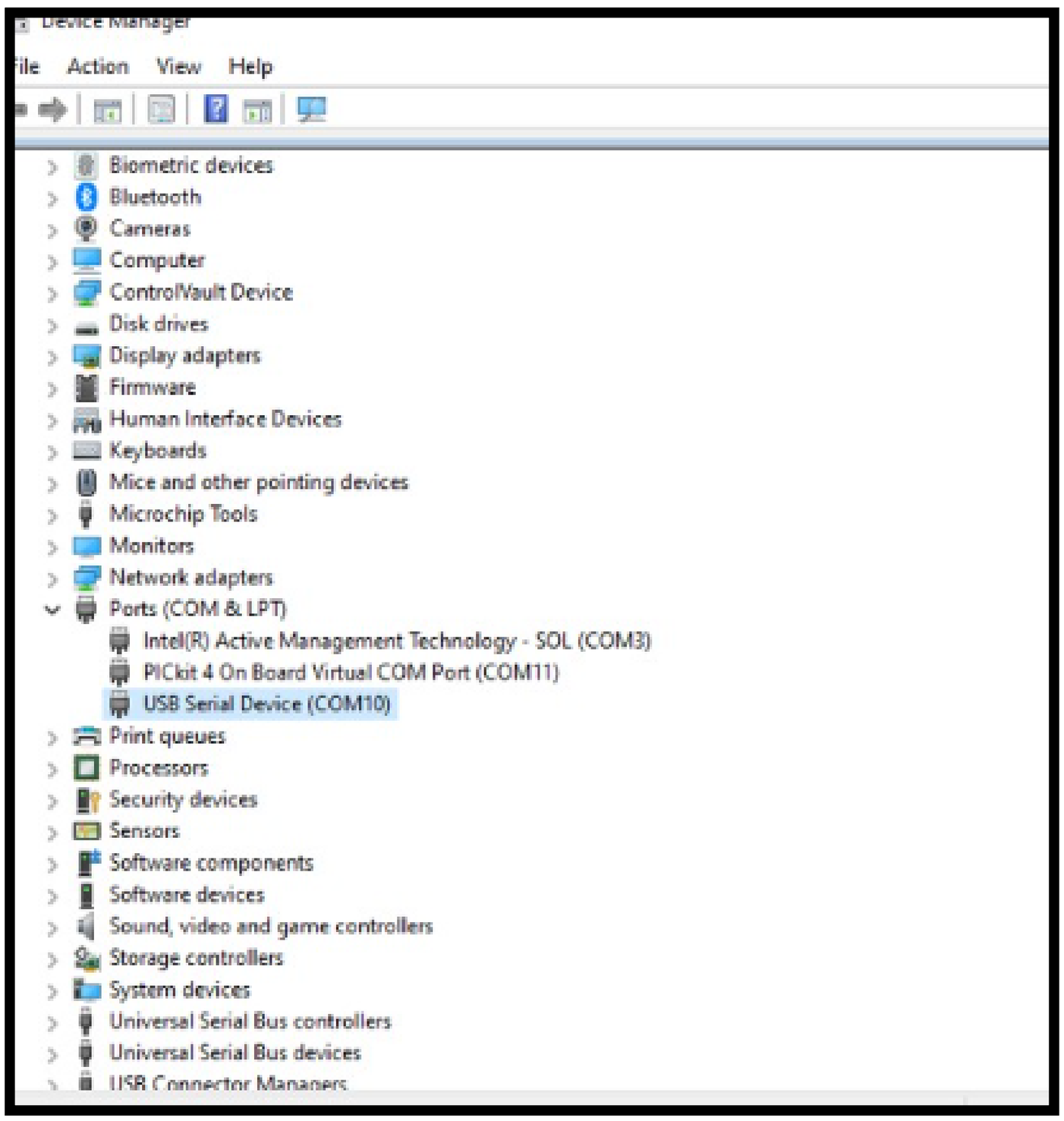
Step 7

Open the content manager nd install the following harmony content.

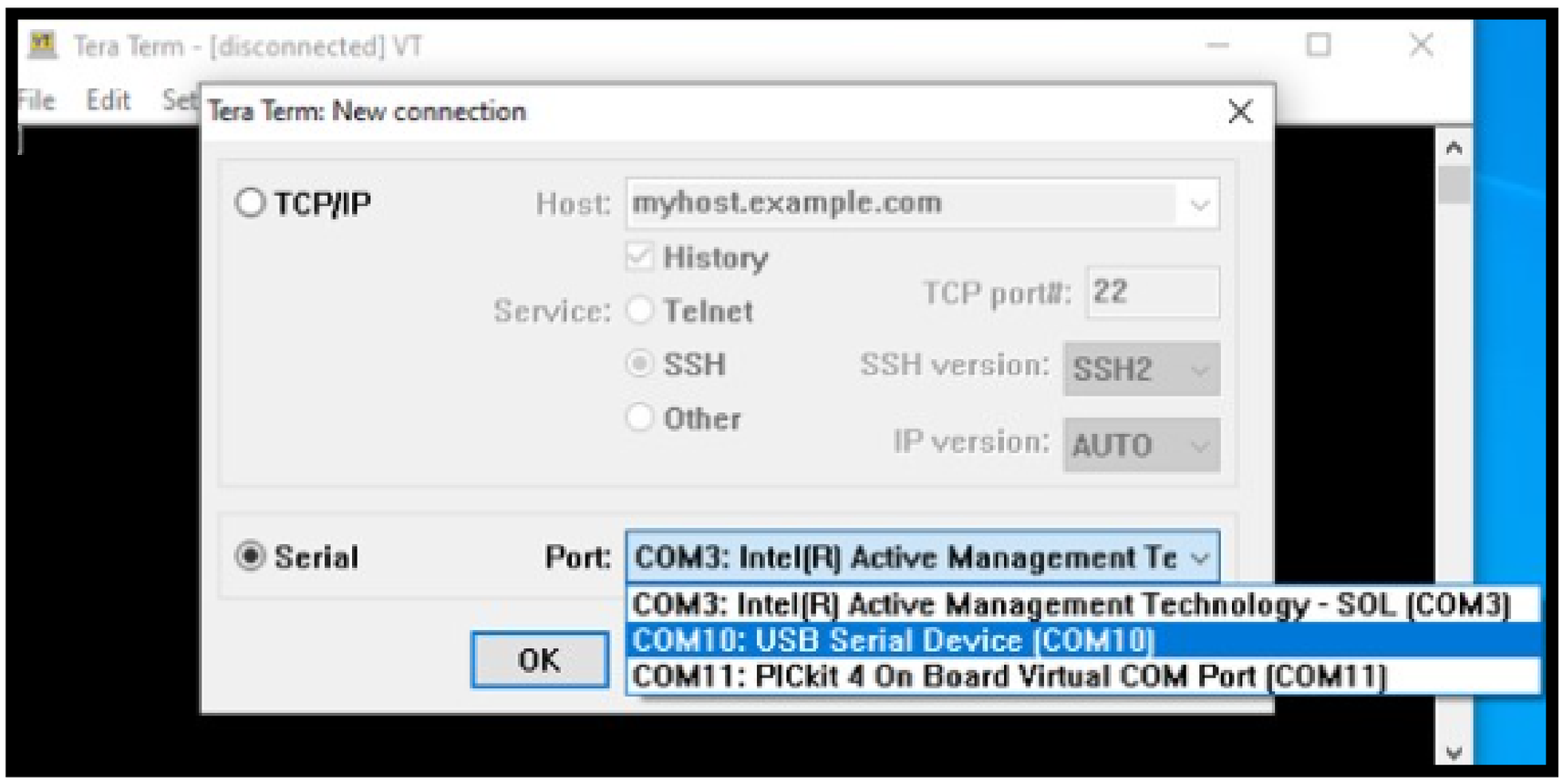


Tera Term

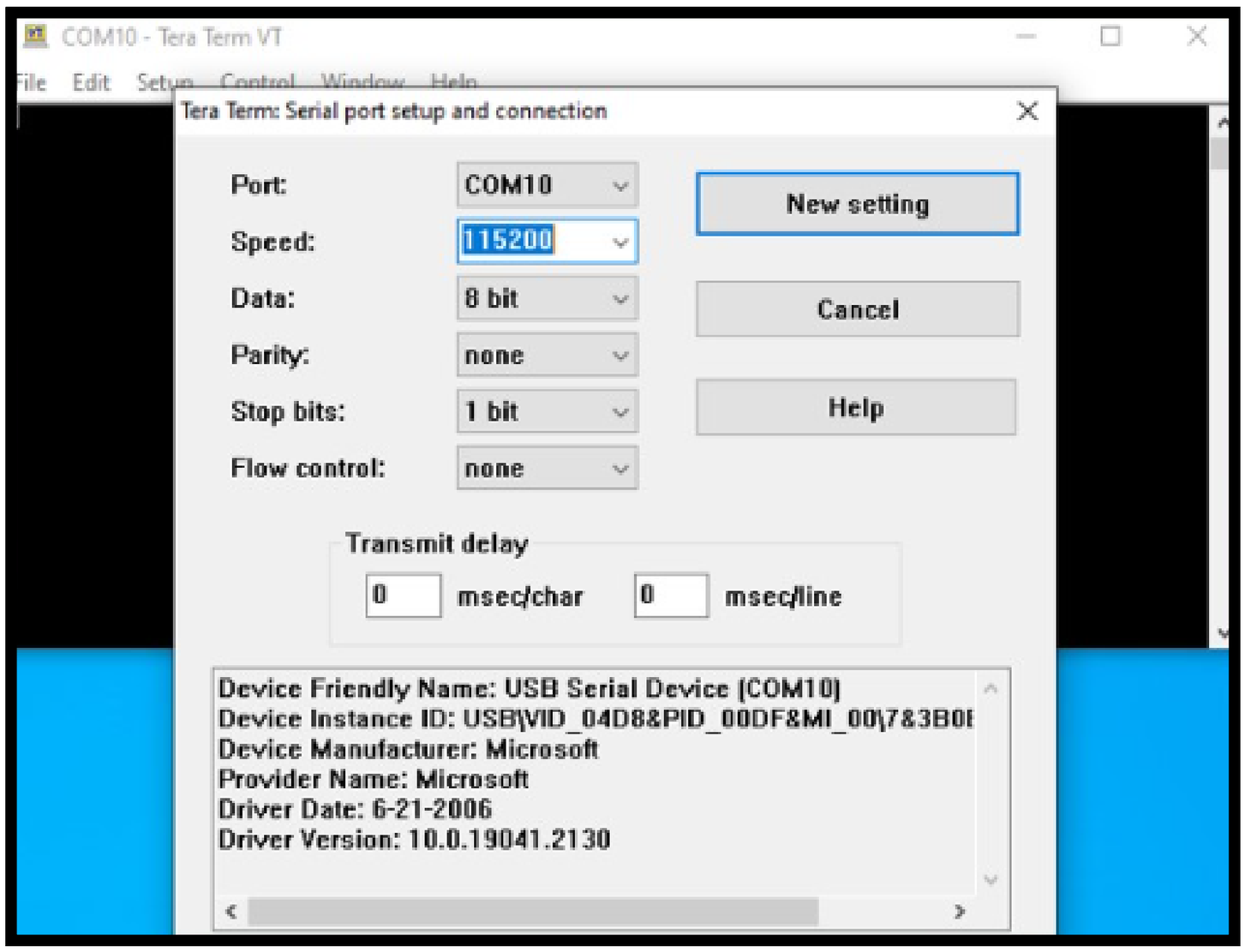
- Download Tera Term: Softonic Tera Term .
- Unplug the Curiosity Board
- Open Device Manager.



- Plug the curiosity board.
- In the tera term Window, select Serial. Select USB serial device from Port.



- In the Setup tab, change the value from 9600 to 115200 in “Speed”. Click New setting to apply the changes

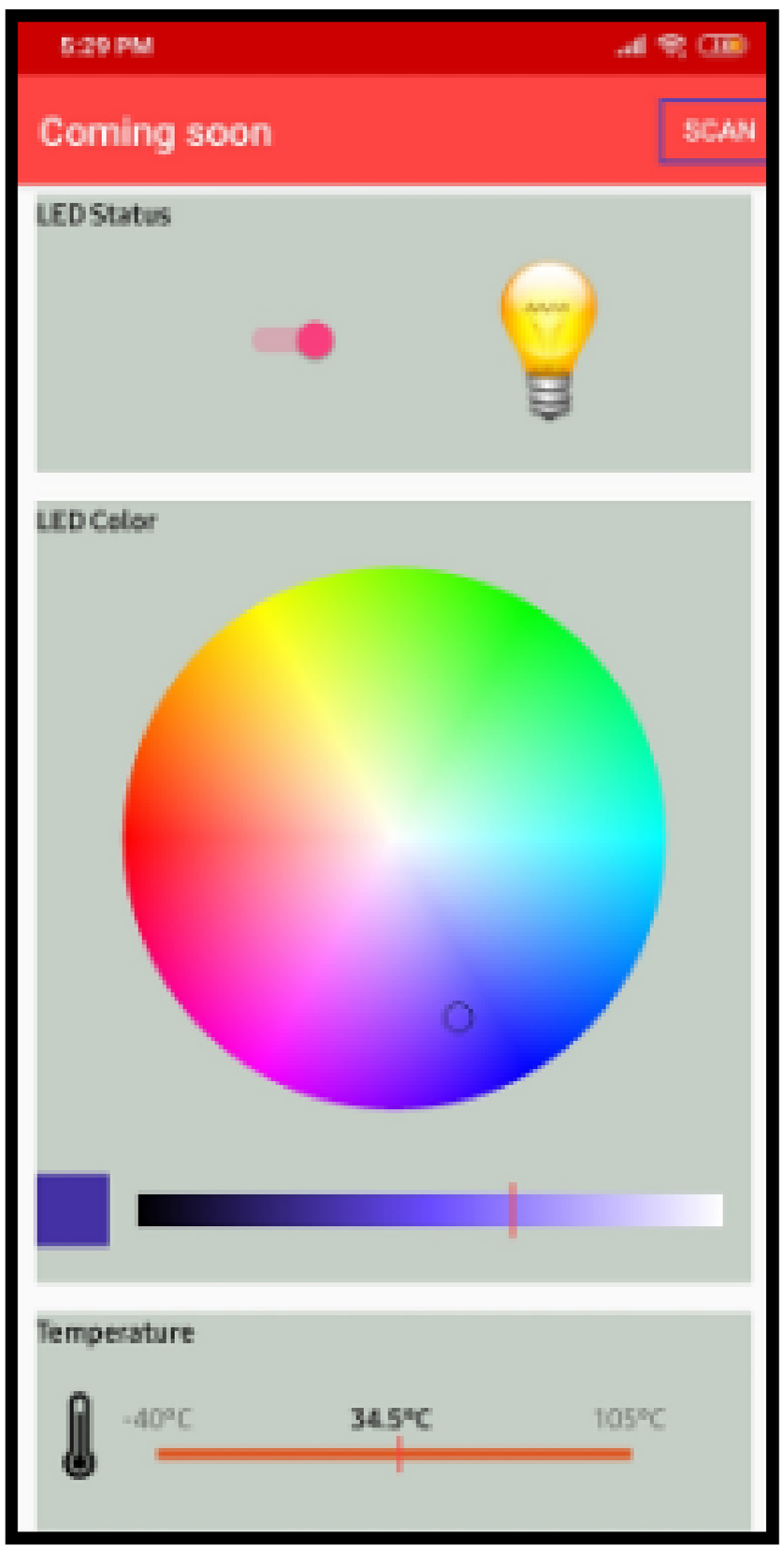


- Press the reset button on the board to see text on the terminal if the curiosity board is programmed to do that.

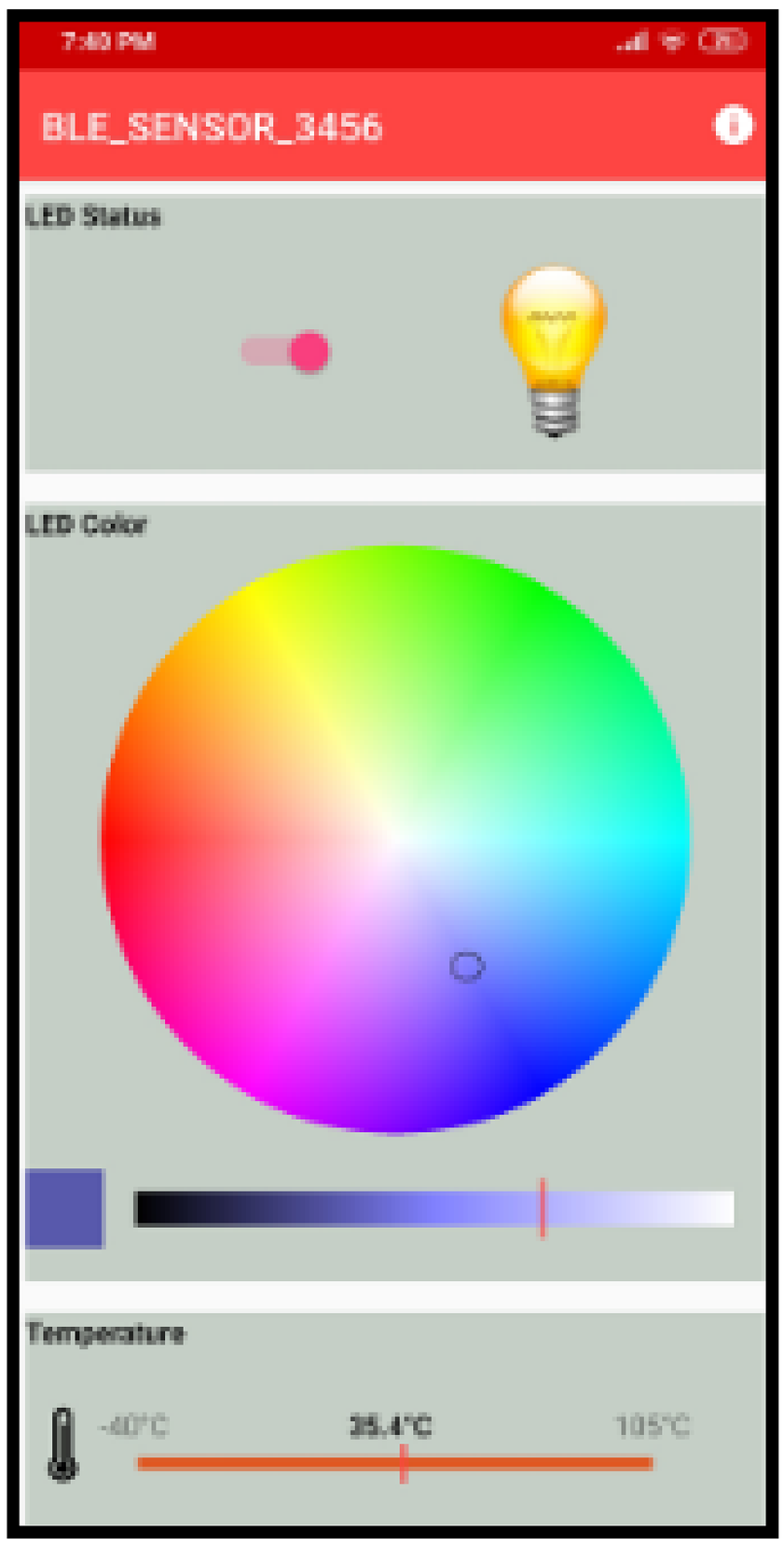
BLE Sensor Project

The WBZ451 module acts as a BLE peripheral, advertising temperature data and RGB LED status on startup. When advertising, the User LED blinks at 500 ms intervals, and once a connection is established with the Microchip Bluetooth Data (MBD) mobile app (“BLE Sensor” sub-app on iOS or Android), the User LED turns solid blue. Through the app’s color wheel, you can turn the RGB LED on or off and select a new color: the device receives HSV values over the TRPS transparent profile and converts them to RGB, adjusting the PWM duty cycle on each LED channel accordingly.

Locally, pressing the onboard User Button toggles the RGB LED: a press turns it on (defaulting to green or the last selected color), and a subsequent press toggles it off or back on. Meanwhile, the module reads its onboard temperature sensor once per second and updates the mobile app whenever the temperature changes by more than 1 °C.



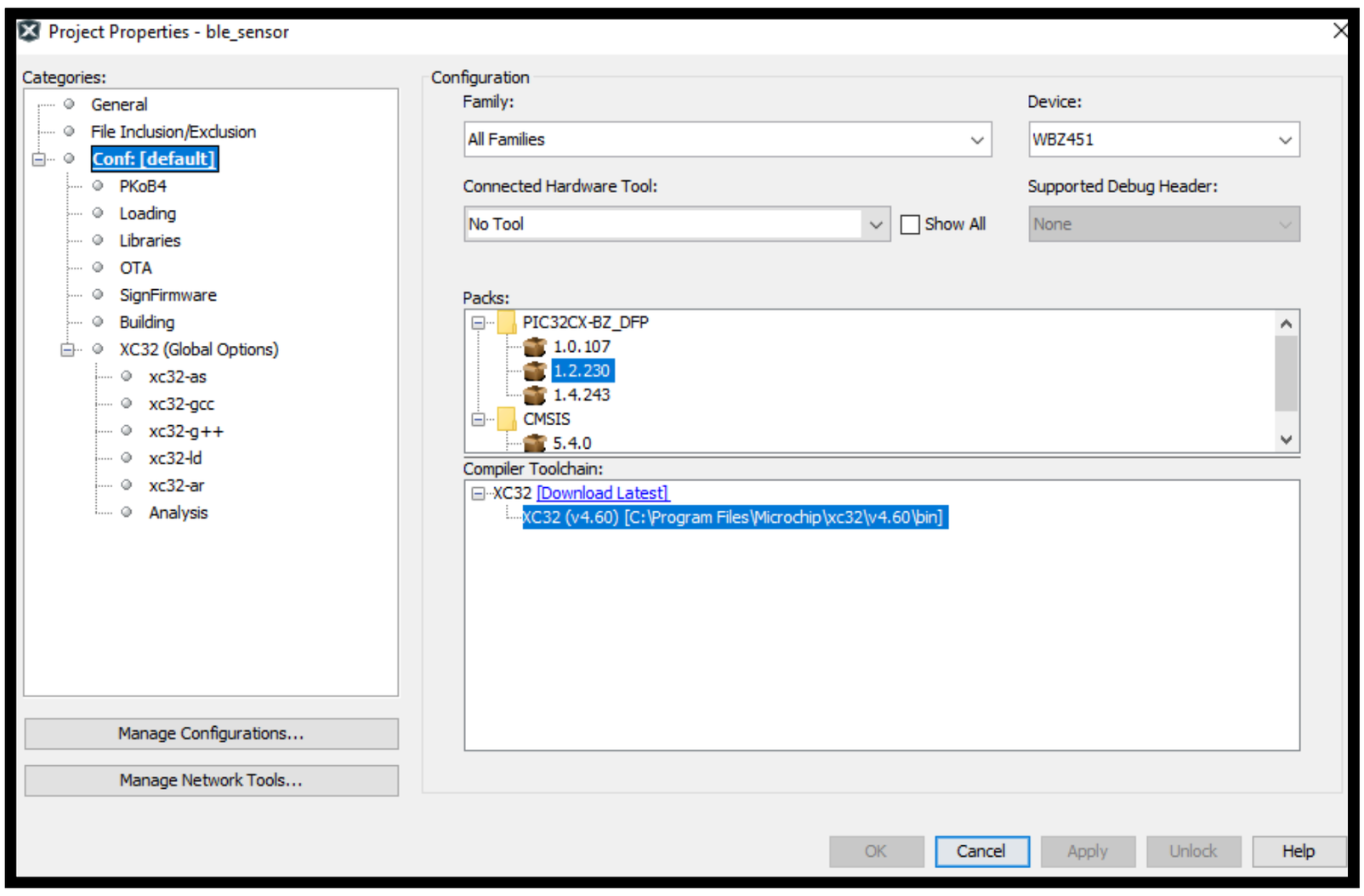
Before connecting to a Device



After connecting to a Device

Steps to Run the Project

- Go to File.
- Click on Open Project
- Navigate to folder where your harmony files are saved.
- Further navigate to wireless_apps_pic32cxbz2_wbz45 – > apps – > ble v– > advanced_applications – > ble_sensor – > firmware – > ble_sensor.X
- Open the project.
- Set the Ble Sensor project as main project.
- Right click on the project to open its properties.

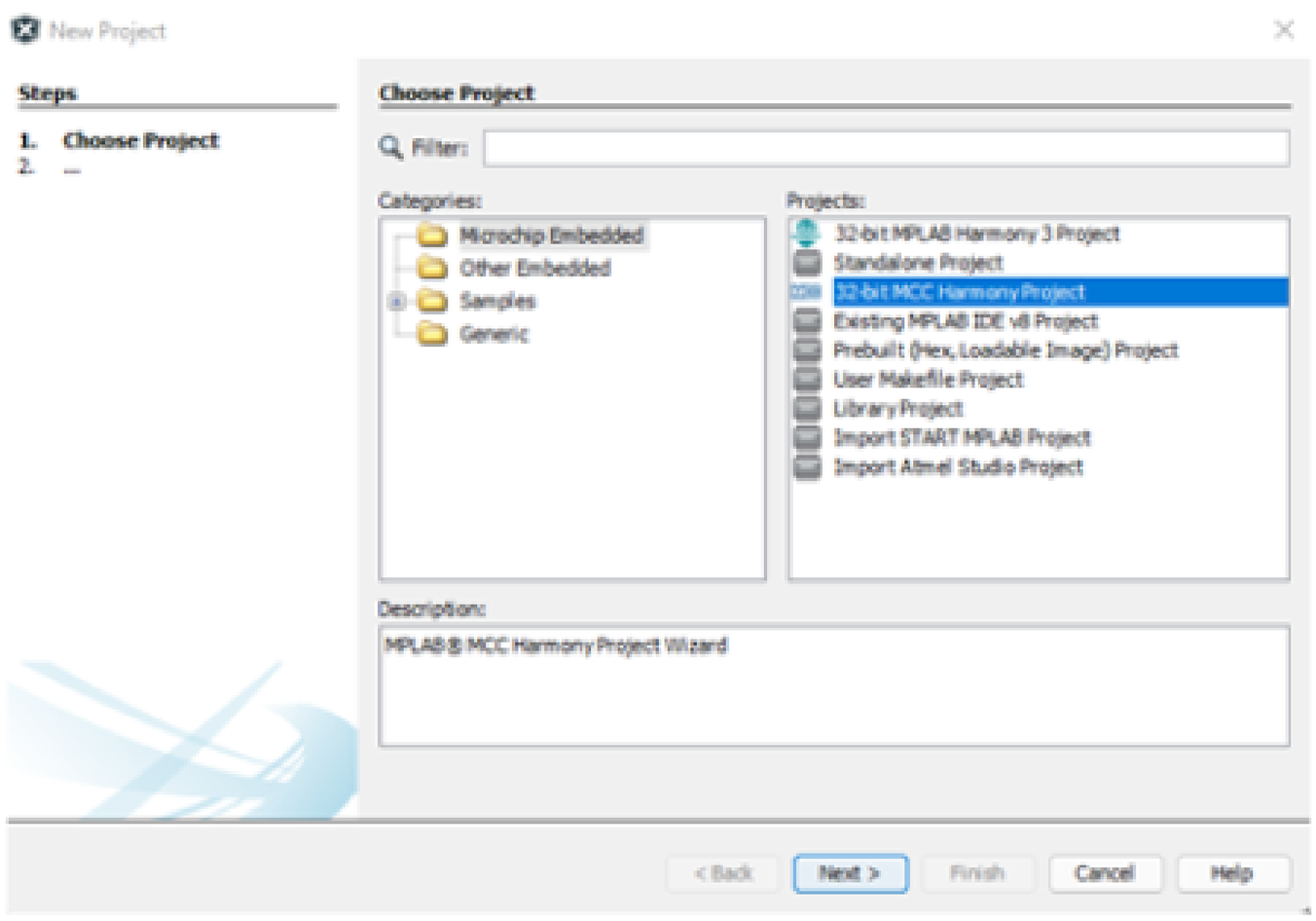


- Select the device packs and compiler.
- Make sure that the device is connected.
- Build the project.
- Run the project.
- Download the Microchip Bluetooth App from Google Play store.
- Select BLE Sensor
- Select PIC32CXBZ
- Click on Scan to scan the available devices.
- Select the available device to connect it to the phone.
- Now you can switch on/off the led, change its color and view the temperature from the app itself.

Problems Faced

- Initially while installing the harmony dependencies we faced various issues in their download.
    - **Solution :** Try installing it multiple items in case such issue occurs.
- If you are following the official webpage of PIC32CXBZ2 Application Developer's Guide, then they have not updated their website accordind to the latest version of MPLAB x IDE.

Figure 2-6. MCC Harmony Project option.



In the latest versions this option to select 32 bit MCC Harmony Project is not available. It is selected by default, you don’t need to select it again.

- **Solution :** Follow the steps mentions in previous pages for software installation.
- Before moving ahead with the BLE Sensor project make sure the dependency **“wireless_apps_pic32cxbz2_wbz45”** is installed.