NXP CUP GUIDE for FRDM-MCXN947

Table of Contents

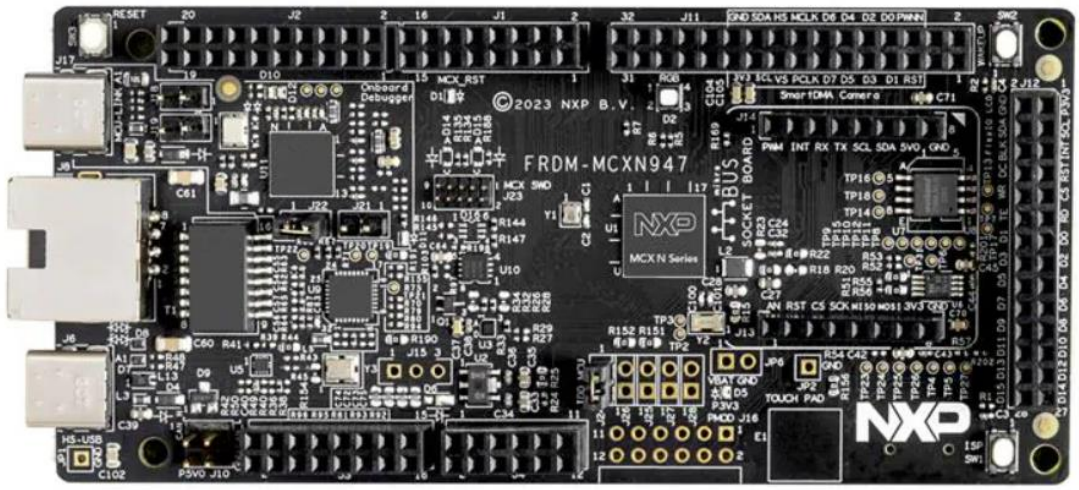
1. Motivation

This project presents an alternative MCU for the *NXP CUP 2024 KIT.*

We wish to share our work on this board, with the other participating students, in hopes of achieving a more united community for this contest 😊.

In this document, you will find an overview of the electrical connections and software.

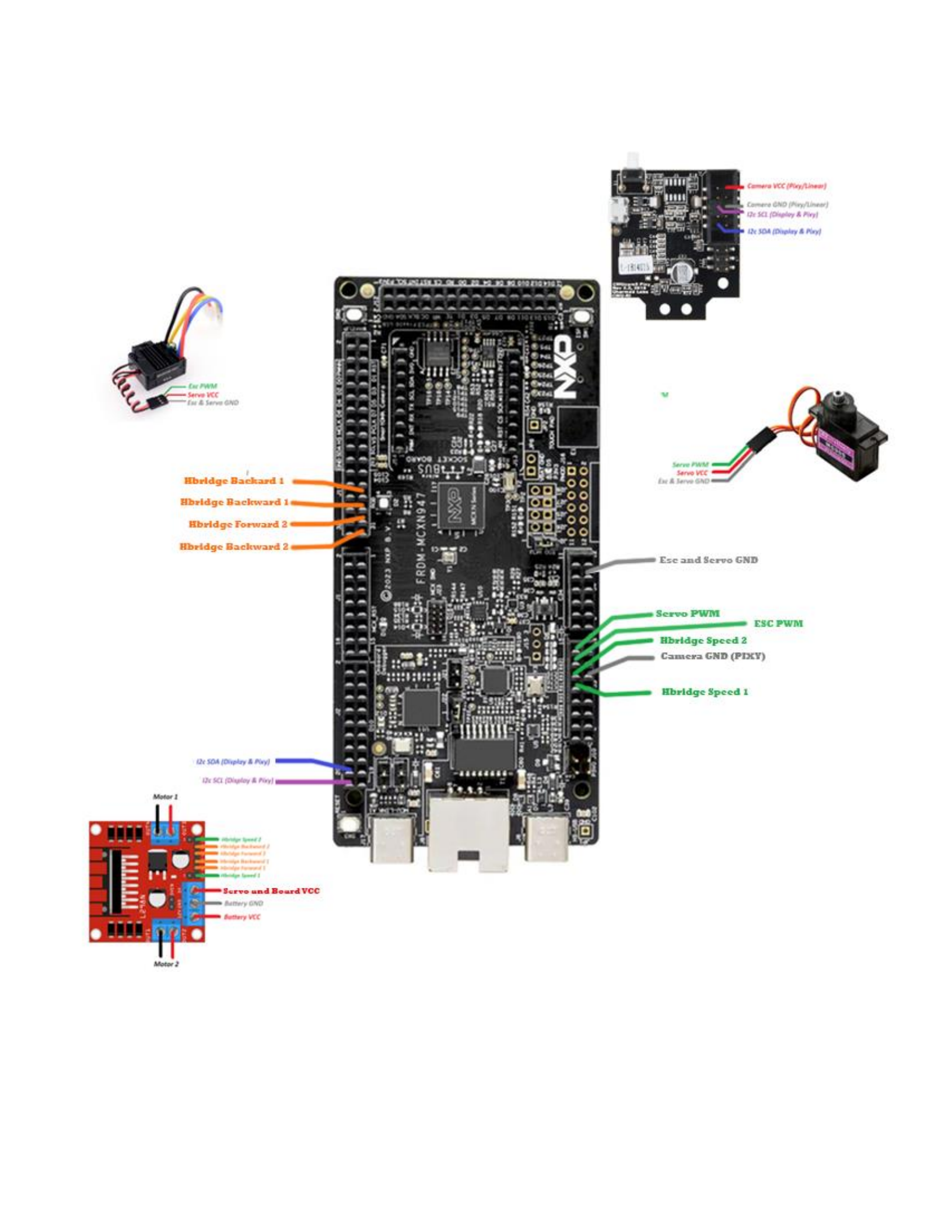
2.Board Overview



1. *Figure 9* ***FRDM-MCXN947***

The **FRDM-MCXN947** is a cheaper alternative to the NXP S32K144EVB board, the FRDM board is around 25eur. You can purchase one from the link below.

MOUSER FRDM-MCXN947 Board


Camera VCC (Pixy/Linear)
Camera GND (Pixy/Linear)
I2c SCL (Display & Pixy)
I2c SDA (Display & Pixy)
Esc PWM
Servo VCC
Esc & Servo GND
Servo PWM
Servo VCC
Esc & Servo GND
Hbridge Backard 1
Hbridge Backward 1
Hbridge Forward 2
Hbridge Backward 2
Esc and Servo GND
Servo PWM
ESC PWM
Hbridge Speed 2
Camera GND (PIXY)
Hbridge Speed 1
I2c SDA (Display & Pixy)
I2c SCL (Display & Pixy)
Motor 1
Servo and Board VCC
Battery GND
Battery VCC
Motor 2

3. Electrical Connections

Component	PORTS/PINS connection
LN298 H-Bridge	P2_4 - ENA P5_8 -IN1 P5_9 -IN2 P1_1-IN3 P2_0 -IN4 P2_5 -ENB +12v-Battery (+) GND-battery (-) 5V- Servo/ESC/frdm-board VIN
Servo Motor	P2_6- Servo pwm H-Bridge&ESC(out) - Servo vcc J2 hearder,14 - Servo & esc ground
ESC	P2_7- Esc pwm H-Bridge&ESC(out) - Servo vcc J2 hearder,14 - Servo & esc ground
Pixy2 Camera	FC4-P1 – Pixy2&Display SCL FC4_P0 – Pixy2&Display SDA - Camera GND - Camera VCC

4. Software

4.1 Installing MCUXpresso IDE

To install MCUXpresso IDE, an account on NXP site must be created. Please create an account at this link.

After the account has been created or if you already have an account on the NXP site, go to the MCUXpresso page found at this link.

The following steps show how to download and install the tool found here

Product Download

MCUXpresso IDE

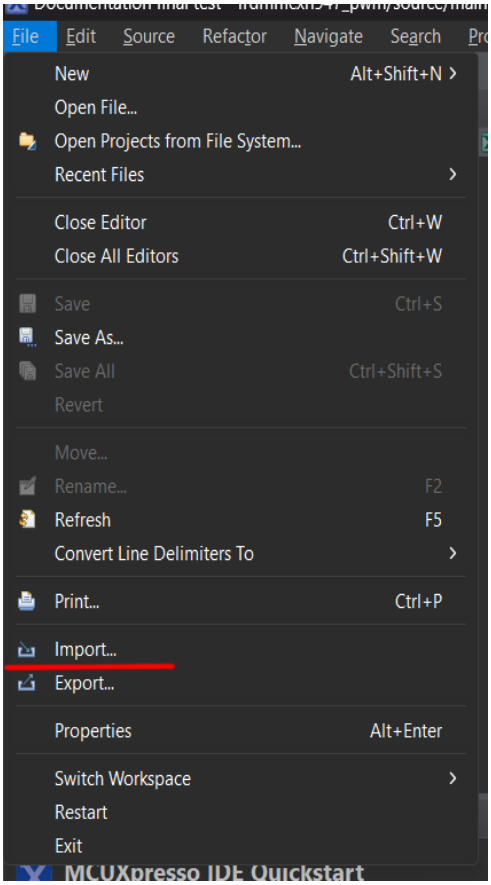
Files | License Keys | Notes — Download Help

Show All Files — 4 Files

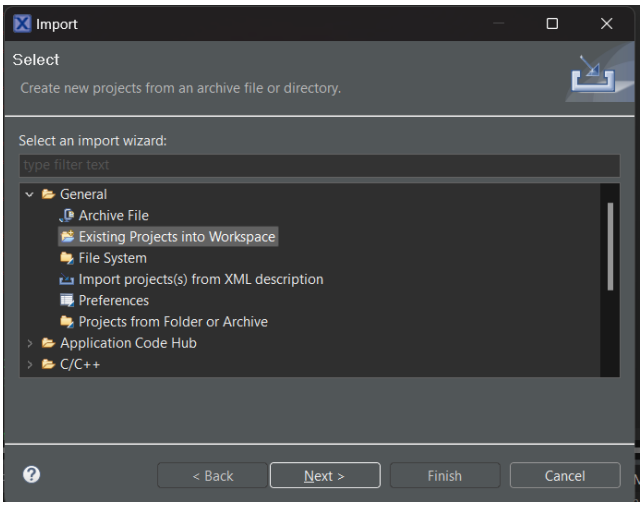
File Description	File Size	File Name
MCUXpresso - Linux - v 24.9	1.4 GB	mcuxpressoide-24.9.25.x86_64.deb.bin
McuXpresso - MAC 86-64	1.1 GB	MCUXpressoIDE_24.9.25.x86-64.pkg
MCUXpresso - Windows v 24.9	1.1 GB	MCUXpressoIDE_24.9.25.exe
MCUXPRESSO_V24.9- Mac AARCH64	1.1 GB	MCUXpressoIDE_24.9.25.aarch64.pkg

4.2 Importing the sample

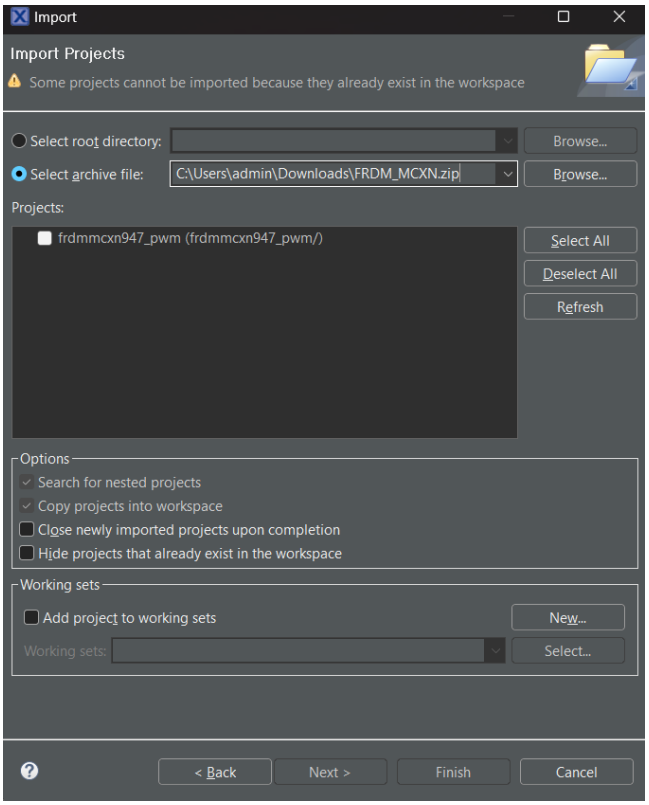
Inside the IDE go to FILE > IMPORT



A window will popup. Go to General > Existing Projects into Workspace



In next popup window, select "*Select Archive File*" and choose the downloaded zip



Select the project and click ***Finish***

To generate the configuration of the drivers, go to Peripherals tab, by clicking the icon indicated by the red arrow. In the Open Perspective popup, select Pins.

4.3 Drivers description and usage

The drivers provided in the example app will be explained in detail in this section, including how they work, how to use them and some improvements you can bring to them to get a competitive advantage during the race.

4.3.1 Hbridge – LN298

This Hbridge controls the power to the motors.

```
void HbridgeSetSpeed(Hbridge *hbridge, int motor1_speed, int motor2_speed)
```

Parameters:

- **motor1_speed –** Values between [-100, 100] are accepted. Negative values cause the motor to go backwards, while positive values direct the motor to go forward. The neutral position is 0.
- **motor2_speed –** same as motor1_speed.

```
int HbridgeGetSpeed(const Hbridge *hbridge, int motor_index) {
    /* Return the last set speed */
    return CarSpeed;
}
```

This function is used to get the Hbridge Speed.

```
void HbridgeSetBrake(Hbridge *hbridge, uint8_t brake_level)
```

This function is used to brake.

```
uint8_t HbridgeGetBrake(const Hbridge *hbridge)
```

This function returns the brake value.

4.3.2 Servo

This driver is designed to control a servo motor via PWM (Pulse Width Modulation) signals. It allows for precise positioning of the servo arm by varying the duty cycle of the PWM signal.

```
void Steer (double angle, int channel)
```

Parameters:

- **angle –** Values between [-100, 100] are accepted. Negative values steer to the left, while positive values steer to the right. The neutral position is 0.
- **channel –** Multiple servos can be used. This parameter indicates the channel that is being affected.

```
void SteerLeft(void);
void SteerRight(void);
void SteerStraight(void);
```

- **SteerLeft:** Moves the servo to its maximum rotation angle (left).
- **SteerRight:** Moves the servo to its minimum rotation angle (right).
- **SteerStraight**: Aligns the servo to its neutral position (center).

4.3.3 Electronic Speed Controller (ESC)

This driver controls the speed and direction of a motor via an Electronic Speed Controller (ESC) using PWM (Pulse Width Modulation) signals. It includes a state machine to handle the motor's operation states, such as Forward, Reverse, Braking, and Neutral.

```
void SetESCPwm(int EscSpeedCommand)
```

Parameters:

- **EscSpeedCommand:** Takes values between [0 and 100] as input. Brake by passing a zero value. Full speed is 100.

```
void SetBrake(uint8 Brake)
```

Parameters:

- **Brake:** if brake has a value, the car will brake.

```
int GetSpeed(void) {
    return currentSpeed;
}
```

This function returns the speed that the car is currently at.

```
uint8 GetBrake(void) {
    return brake;
}

EscStates GetEscState(void) {
    return escState;
}
```

Getbrake returns the current brake value.

GetEscState returns the current state of the ESC.