EE 393
Spring 2020
Jonathan Do
Writing Assignment 1

How to Set-Up CAN Communications For a Teensy 4.0 Microcontroller

Prepared by: Jonathan Do, Electrical Team Lead of Washington Superbike

Last Revised: May 2020

Objective

The purpose of this document is to provide instructions on setting up the CAN bus for use on a Teensy 4.0 microcontroller. This manual is written for new electrical team members of Washington Superbike who are interested in firmware development. This procedure is followed to help members take their first steps in firmware development for electric vehicles. This procedure is to be done after a motor controller is selected or identified and concurrently with assembling the electric powertrain. These step-by-step instructions on setting up the CAN bus are reliant on circuit construction, and embedded C programming using the Arduino integrated development environment (IDE).

Background

This section provides the list of materials and the knowledge needed to set up the CAN bus on the Teensy 4.0 as well as cautions/warnings that need to be accounted for.

Required Items

- Teensy 4.0
- Arduino IDE
- 3.3V CAN transceiver
- ~5 jumper wires for breadboarding

Relevant Concepts

For this project, the user will need to use embedded C programming, circuit construction, and CAN bus communications. Familiarity with the Arduino IDE is recommended.

The Teensy 4.0 is a microcontroller that is programmed in embedded C using the Arduino IDE. This microcontroller has many applications due to the various I/O that it has. In this case, we will be using the CAN ports on the Teensy for our CAN communications.

The Arduino IDE is a popular platform for programming development microcontrollers. The IDE uses C/C++ as its programming language and compiles the code into assembly. A “setup” and “loop” function is provided by the IDE which allows the user to simply type in code that would be run initially on startup and continuously on a loop.

Embedded C is an extremely popular language for programming microcontrollers that is used for almost every single electrical device including phones, televisions, and smart watches.

The Controller Area Network bus, or CAN bus, is an electrical communication protocol that is popular in automotive applications. The CAN bus works by having two voltage lines, one CAN high signal (CAN_H) and one CAN low signal (CAN_L). The CAN_H line hovers at a higher voltage and the CAN_L line hovers at a lower voltage. The difference in voltage in the CAN_H and CAN_L line while hovering is translated as a logical “0” and is known as the dominant voltage. A logical “1” occurs when the two lines, CAN_H and CAN_L, are brought together to the middle so that the voltage difference between CAN_H and CAN_L is close to 0. This is known as the recessive voltage.

The baud rate is the rate at which bits are transmitted across a communication bus. The baud rate is measured in bits per second so, for example, a baud rate of 9600 would mean that the communication bus is transmitting 9600 bits per second. It is vital that every device on the communication bus is running at the same baud rate so that no bits are missed or read twice.

Every package sent on the CAN bus has a unique CAN ID in which it can be identified by.

Cautions and Warnings

- Ensure that the Teensy 4.0 pins are not handling more than 3.3V. Higher voltage will risk frying the microcontroller
- Always check for shorts before applying power
- Avoid spilling liquid onto the set up, this might cause a short
- Always make sure power is completely disconnected before adding or removing anything to the circuit

Procedure

This procedure can be broken down into four stages: setting up the circuit, installing the Arduino IDE, setting up the Arduino IDE for use with CAN, and programming.

Setting Up the Circuit

1. Refer to this pinout of the Teensy 4.0 and select a pair of CRX and CTX pins to use for your CAN port. For this example, we will use pins 22 and 23 (CTX1 and CRX1). The pinout

for the Teensy 4.0 is shown below (Fig. 1):

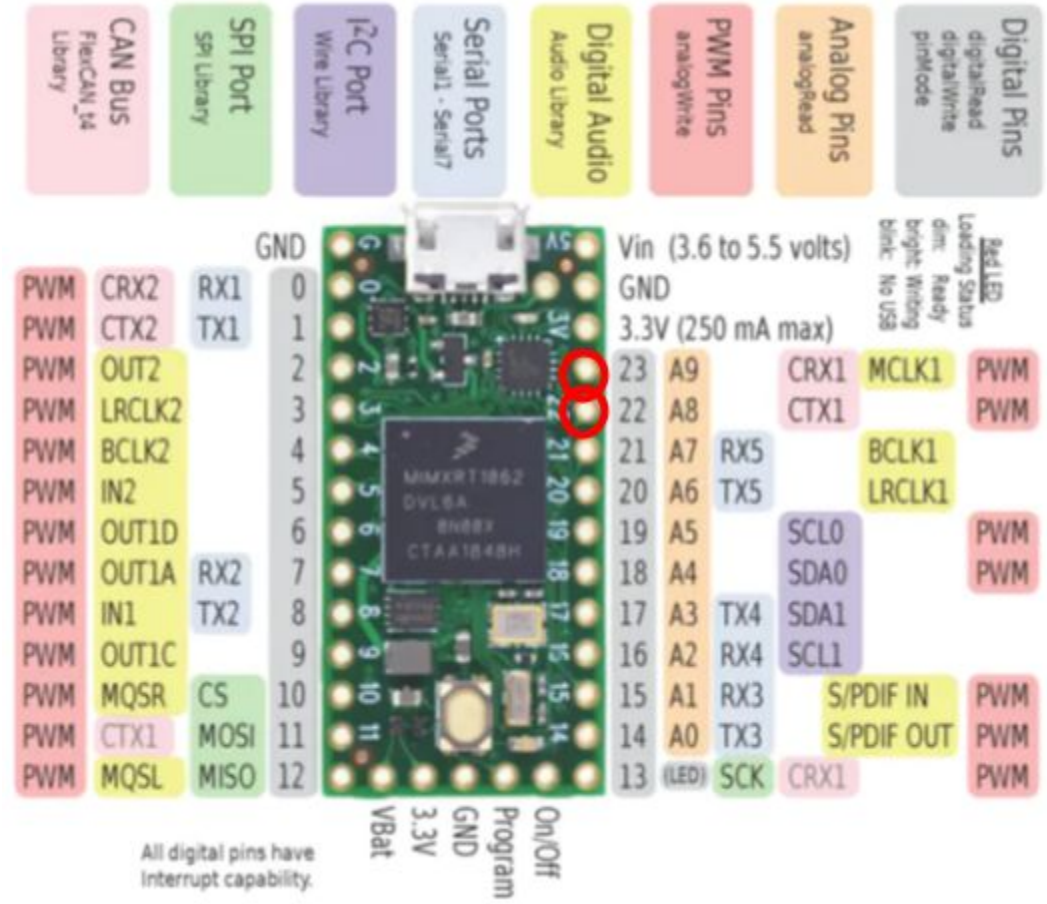

Fig. 1. The pinout of a Teensy 4.0.

2. Connect the chosen CTX pin on the Teensy to the CAN transceiver's TX pin and the chosen CRX pin on the Teensy to the CAN transceiver's RX pin.
3. Connect the CAN transceiver's 3.3V pin to the Teensy's Vin pin.
4. Connect the CAN transceiver's GND pin to the Teensy's GND pin.
5. Connect the CAN transeiver's CAN_H and CAN_L pins to the external CAN_H and CAN_L lines, respectively.

Installing the Arduino IDE

1. Visit https://www.arduino.cc/en/Main/Software and download the appropriate Arduino IDE version for your operating system. For this guide, we will be using Windows 10.
2. Run the installer.

3. Left click “I Agree” at the License Agreement screen show below (Fig. 2):

Fig. 2. The start-up screen of the Arduino IDE installer.

4. Make sure all the boxes are checked at the Installation Options screen. Then left click “Next.” The Installations Options screen should look like the one below (Fig. 3):

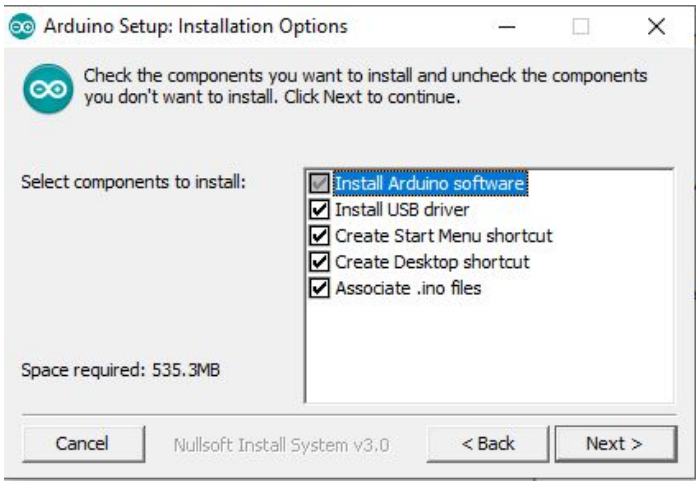

Fig. 3. The installation options screen of the Arduino IDE installer.

5. Left click “Install” at the next screen and wait for the installation to finish.
6. Left click “Close.”

Setting Up the Arduino IDE for use with CAN

First, we will be installing the Teensyduino library:

1. Download and run the installer for the proper operating system: https://www.pjrc.com/teensy/td_download.html
2. Left click “Next” until the Libraries to Install screen .
3. Ensure that “All” is selected. Then, left click “Next.” Fig. 4 below shows what the windows should look like:

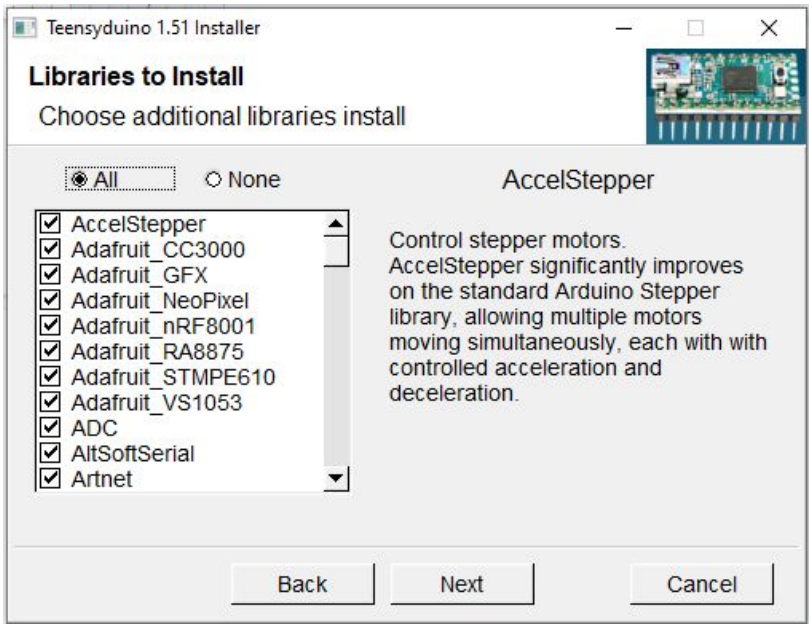

Fig. 4. Library selection screen of the Teensyduino installer.

4. Left click “Install” and wait for the program to finish installing.
5. Take note of the final screen (Fig. 5). This will be a good reference when uploading the firmware to the Teensy:

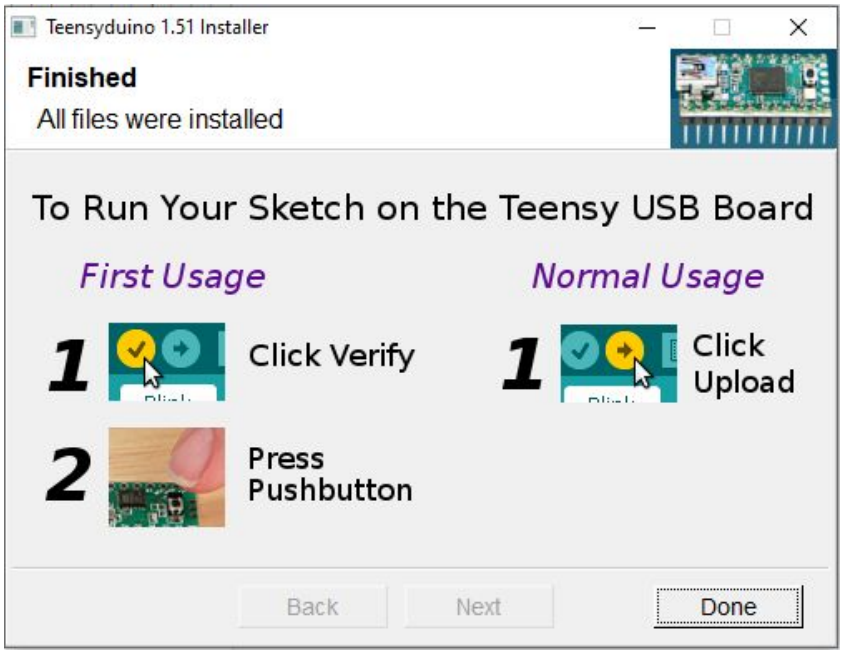

Fig. 5. The final screen of the Teensyduino installer.

6. Left click “Done.”

Next, we will set up the Arduino IDE for use with the Teensy 4.0:

1. Open up the Arduino IDE:
2. Left click the “Tools” tab in the top left corner (Fig. 6):

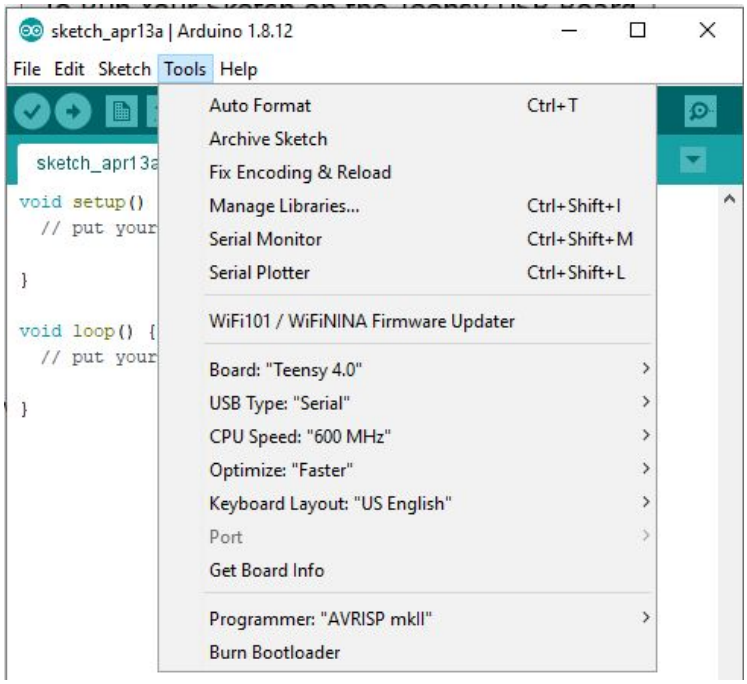

Fig. 6. Tools tab of the Arduino IDE.

3. Highlight “Board” with your cursor and make sure “Teensy 4.0” is selected which is indicated by the large dot (Fig. 7):

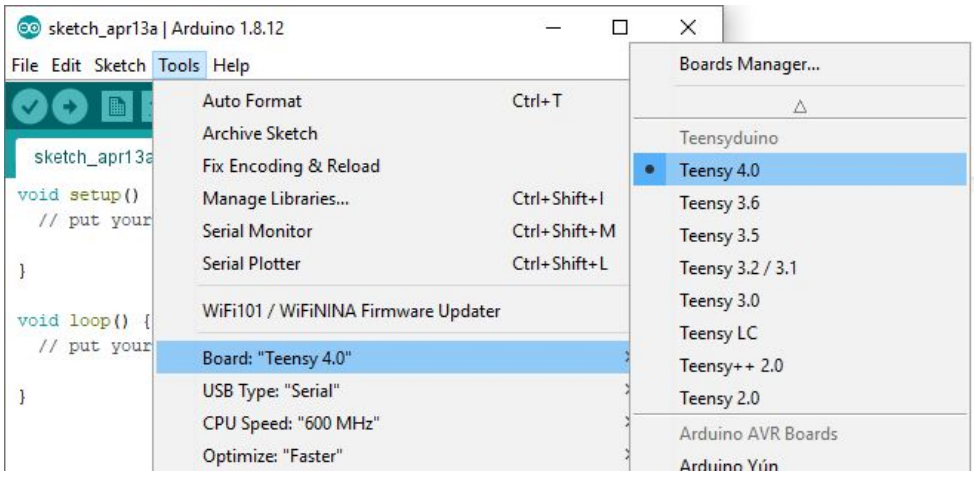

Fig. 7. Selecting the Teensy 4.0.

Programming

Before typing in actual code, we must import the FlexCAN_T4 library into our project:

1. Navigate using the top left toolbar to Sketch->Include Library.

2. Find and left click “FlexCAN_T4 (Fig. 8)”:

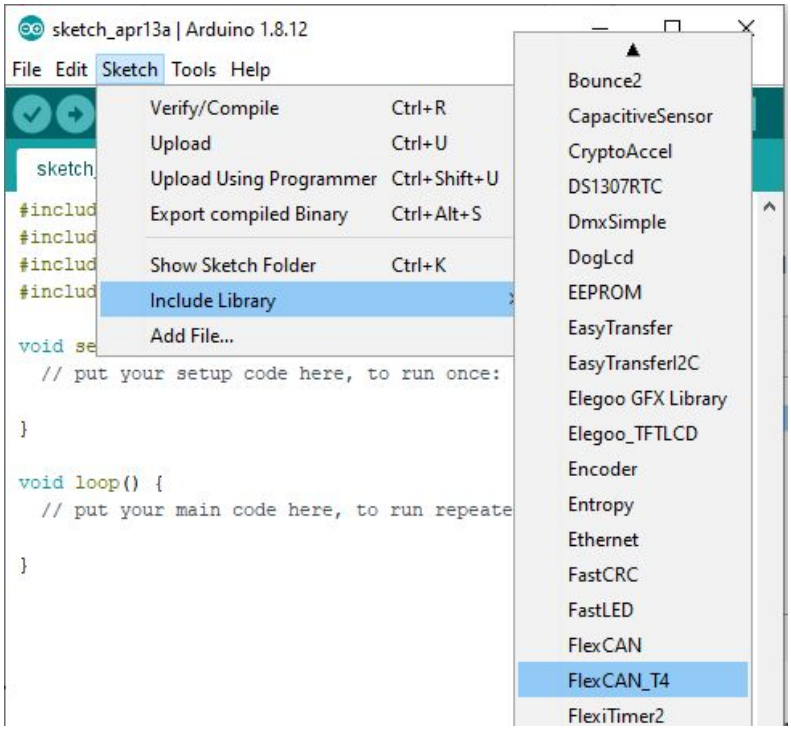

Fig. 8. Selecting the FlexCAN_T4 library to be included.

FlexCAN_T4 is a version of the FlexCAN library specifically designed for the Teensy 4.0.

3. If successful, the top of the sketch should have the follow statements (Fig. 9):

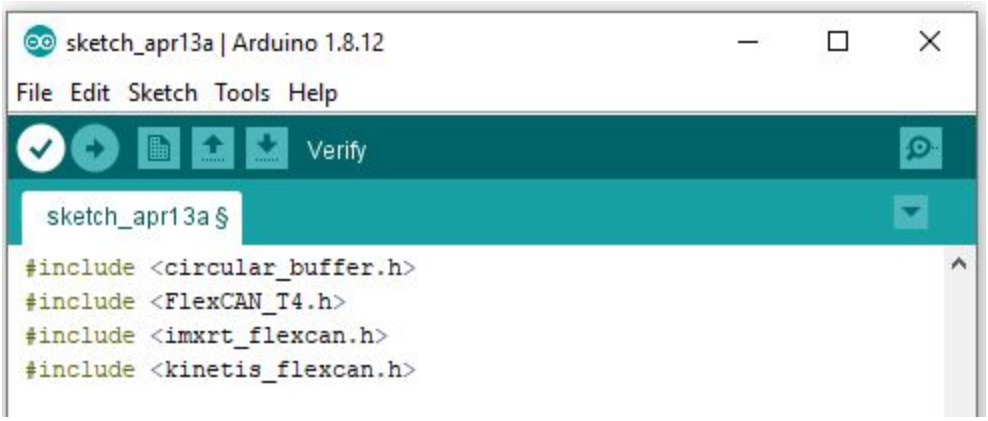

Fig. 9. The inclusion of the correct libraries.

Finally, we will initialize the CAN bus:

1. Create a new global FlexCAN_T4 object with the pin numbers chosen when setting up the circuit and create a global CAN_message_t to store the CAN messages (Fig. 10) [1][2]:

```
#include <circular_buffer.h>
#include <FlexCAN_T4.h>
#include <imxrt_flexcan.h>
#include <kinetis_flexcan.h>

FlexCAN_T4<CAN1, RX_SIZE_256, TX_SIZE_16> CAN_BUS;
CAN_message_t CANmsg;
void setup() {
  // put your setup code here, to run once:

}
```

Fig. 10. Preparing global variables.

2. Figure out the baud rate that your CAN bus is using. This can be done by checking the documentation of other devices that you may have on the CAN bus. In this example, we use a baud rate of 250000.
3. In the setup() function, use begin() [1][2] to begin CAN communications and set the baud rate to what your CAN bus requires (Fig. 11):

```
void setup() {
  // put your setup code here, to run once:
  CAN_BUS.begin();
  CAN_BUS.setBaudRate(250000);
}
```

Fig. 11. Initialization of the CAN bus.

4. Now you are ready to start sending and receiving CAN messages using the Teensy 4.0

References

[1] A. Brewer. FlexCAN_T4. (2019) [Online]. Available: https://github.com/tonton81/FlexCAN_T4

[2] Pawelsky. FlexCAN_Libarary. (2019) [Online. Available: https://github.com/pawelsky/FlexCAN_Library