Team T20 – UTS DMMS Autumn 2023

Jeremy.j.mansfield@student.uts.edu.au

Abstract

This document contains a description and explanation of how the Arduino IDE Code for the   
Team T20 Automotive Air Pressure Logger functions.

Automotive Air  
Pressure Logger

CODE DOCUMENTATION & USERS MANUAL

[What is the Automotive Air Pressure Logger? 2](#_Toc135149281)

[Usage Instructions 2](#_Toc135149282)

# What is the Automotive Air Pressure Logger?

The Automotive Air Pressure Logger is an aerodynamic analysis solution designed by Team T20 for the UTS Motorsport Electric Team to assist in the verification of CFD Simulation Results.

# Usage Instructions

1. Open the enclosure and insert a FAT32 Formatted MicroSD Card into the SD Card on the microcontroller board.
2. Close the enclosure.
3. Power on the device.
4. Wait for the device to complete its initialization process and arrive to the menu.
5. Press the “start” button to open the run menu.
6. Press “start” again to begin logging. The display should display the run-time of the current logging session.
7. When you have finished logging, press the “start” button again to complete the session.
8. Wait for the device to finish processing the data.
9. Turn off the device and remove the SD card.
10. Insert the SD Card into a computer and use Matlab or a similar program to analyse the data.
11. <To be continued when/if matlab code is developed>

# Code Documentation

## SD Card

### Pre-requisites

To use any of the SD Card functionality, the following libraries must be installed and included in any relevant code:

* Teensy Support
* Teensy SD Card Library (SD.h) - <https://github.com/PaulStoffregen/SD>

### Functions

#### SD Card Library Initialization

* SD.begin(BUILTIN\_SDCARD);
  + The BUILTIN\_SDCARD variable is included in the Teensy SD Card Library. If an external SD Card Module is used, this variable must be replaced with the relevant value.
  + Must be performed prior to usage of any other SD Card Functionality.
  + Returns a “bool”:
    - False if card is not valid or no card is installed.

#### SD Card File Read

* SD.open(document, accessmode);
  + The FILE\_READ variable is included in the Teensy SD Card Library.
  + “document” is a String of the file name and/or address of the document being read. Normally this is a .txt or similar file type.
  + accessmode is either FILE\_WRITE or FILE\_READ depending on whether the file is to be written to or read from.
  + Returns a “File” object of the file at “document”.
  + Example usage:
    - File dataFile = SD.open(“datalog.txt”, FILE\_READ);
* file.read();
  + “file” is a “File” object that has had a file assigned to it using SD.open(document, FILE\_READ);
  + Returns the data in the “File” object variable and returns it as a string [CONFIRM].
  + Example usage:
    - Serial.write(dataFile.read());
* file.close();
  + “file” is a “File” object that has had a file assigned to it using SD.open(document, FILE\_READ);
  + Closes “file”.
  + Example usage:
    - dataFile.close();

#### File Availability Check

* file.available();
  + “file” is a “File” object that has had a file assigned to it using SD.open(document, FILE\_READ);
  + Returns a “bool” value:
    - True if the file is available to be written to
    - False if file is not available for writing
  + Normally used within a loop to read/write only while the document is available for writing or reading and fail if it becomes unavailable.
  + Example usage:
    - While(dataFile.available()){

Serial.write(dataFile.read());

}

#### Write to File

* file.print(text);
  + “file” is a “File” object that has had a file assigned to it using SD.open(document, FILE\_WRITE);
  + “text” is a string that is to be written to the latest line of the file.
  + Example usage:
    - dataFile.print(“This is the data to be written.”);
* file.println(text);
  + “file” is a “File” object that has had a file assigned to it using SD.open(document, FILE\_WRITE);
  + “text” is a string that is to be written to the latest line of the file, followed by the addition of a new line.
  + Example usage:
    - dataFile.println(“This is the data to be written before a new line.”);

## Multiplexer

[To be completed once the multiplexer has arrived]

## Sensor Usage

The Automotive Air Pressure Logger uses the MS5637 sensor to measure air pressure, and addresses individual sensors over I2C through a multiplexer, due to the MS5637’s hardware-set I2C Slave Address.

### Pre-Requisites

To take sensor readings from the MS5637 sensor, the following libraries are required:

* Teensy Wire (i2c) Library (Wire.h)
* Sparkfun MS5637 Arduino Library (SparkFun\_MS5637\_Arduino\_Library.h)

### Functions

#### I2C Initialization

* WireX.begin();
  + X is the number/channel of the i2c interface being used. In this logger, Wire1 is used for the first sensor string.
  + Initializes the i2c on the specified channel.
  + Must be performed prior to using any other i2c-related functionality.
  + Example usage:
    - Wire1.begin();

#### Sensor Initialization

* MS5637 barometricSensor;
  + Creates an MS5637 object to communicate with the sensor.
  + Example usage:
    - MS5637 barometricSensor;
* barometricSensor.begin(WireX);
  + Initializes the sensor on i2c channel/wire X.
  + Must be performed prior to using any other sensor-related functionality.
  + Returns a bool value:
    - True if sensor is successfully started.
    - False if sensor fails to respond.
  + Example usage:
    - barometricSensor.begin(Wire1);

#### Read Sensor Pressure

* barometricSensor.getPressure();
  + Returns pressure read from the sensor as a float value in hPa/mBar.
  + Example usage:
    - barometricSensor.getPressure();

#### Read Sensor Temperature

* barometricSensor.getTemperature();
  + Returns temperature read from the sensor as a float value in Celsius.
  + Example usage:
    - barometricSensor.getTemperature();

### Addressing the Sensor Through the Multiplexer

For detail on addressing different sensors through the multiplexer, see the “Multiplexer” section.

## Menu Mode State Machine

[Complete once code is finished. Include state machine diagram and debouncing notes]