CanSat 2020-2021 — Team Autonomeasure — Can code documentation

Joep van Dijk

December 28, 2020

Table of contents

1 Can overview

1.1 Modules and sensors in the Can

The micro controller in the Can is an Arduino Mega 2560. There are a few modules/sensors that will be connected to the Arduino.

- The APC220 radio module, it is used to transmit and receive data.
- The Adafruit BMP280 module, it measures the temperature and pressure.
- The Adafruit MPU6050 module, it is an accelerometer and gyroscope, it also measures the temperature.
- The NEO-6M GPS module which receives GPS data.
- The HW-125 SD card module, it is used to save all the radio transmissions (and maybe some additional data).

1.1.1 The APC220 radio tranceiver module

The APC220 radio tranceiver module will be connected to the rx1 and tx1 serial pins.

1.1.2 The Adafruit BMP280 module

The BMP280 module is connected to the I^2Cpins on the Arduino. The Adafruit BMP280 library[1] uses the I^2Cprotocol to communicate with the BMP280 module.

1.1.3 The Adafruit MPU6050 module

The MPU6050 module is connected to the I^2Cpins on the Arduino. The Adafruit MPU6050 library[2] uses the I^2Cprotocol to communicate with the MPU6050 module.

1.1.4 The NEO-6M GPS module

The NEO-6M GPS module is connected with the rx2 and tx2 serial pins.
Because the NEO-6M does not handle 5V well a simple voltage level converter so the 5V signal from the Arduino is converted to a 3.3V signal before it reaches the GPS module.

1.1.5 The HW-125 SD card module

The HW-125 SD card module is connected with the `SPI` pins, on the Arduino Mega that are the pins: `D50`, `D51` and `D52`. The default `SD` library is used to communicate with the module. All the radio transmissions will be saved on the SD card so if any of the radio transmissions fail we can retrieve them later.

[1] Adafruit BMP280 library: https://github.com/adafruit/Adafruit_BMP280_Library
[2] Adafruit MPU6050 library: https://github.com/adafruit/Adafruit_MPU6050_Library

2 All the custom structures

2.1 Error

The Error structure contains information about the error that has occurred. The property `errorID` is a `uint8_t` that contains the identifier of the error. All the possible identifiers are:

1. An error occurred while initializing the BMP280 module, `Adafruit_BMP280::begin` returned *false*.
2. An error occurred while initializing the MPU6050 module, `Adafruit_MPU6050::begin` returned *false*.
3. The APC220 was not configured correctly.
4. An invalid GPS time was received.
5. An invalid GPS location (latitude and longitude) was received.
6. An invalid GPS altitude was received.

255. An unknown error has occurred.

2.2 Vector3

The Vector3 structure is a vector with three axis (x, y and z). All the properties have the type `float`.

3 All the custom classes

3.1 GPS

The GPS class contains some methods that do calculations (e.g. a method to calculate the vertical velocity).

3.1.1 void GPS::GPS(HardwareSerial *gpsSerial)

Arguments

- `gpsSerial` is a pointer to a `HardwareSerial` object (e.g. `Serial2`) used to communicate with the GPS module.

Returns

`void`

Description

This is the constructor of the class. It sets the `gpsSerial` variable.

Scope

`PUBLIC`

3.1.2 void GPS::begin(uint8_t baudrate = 9600)

Arguments

- `baudrate` is an `uint8_t` and holds the baudrate that is going to be used for the serial connection with the GPS module. The default value is `9600`.

Returns

`void`

Description

Call the begin function of the `gpsSerial` object and sets the UART baudrate.

Scope

`PUBLIC`

3.1.3 void GPS::read(void)

Arguments

- void

Returns

`void`

Description

Read the incoming serial data from the GPS module and encodes it with `TinyGPSPlus::encode`.

Scope

`PUBLIC`

3.1.4 uint8_t GPS::get_time(Error *errors, char *time)

Arguments

- `errors` is a pointer to an `Error` object, all the errors that occur will be saved to this pointer
- `time` is a pointer to a `char` array with four elements containing the time.

Returns

`uint8_t` - The amount of errors that occurred.

Description

Get the current time and save it to the `time` array.
The first element will contain the hour (0-23), the second element will the minutes (0-59), the third element will contain the seconds (0-59) and the fourth element will contain the centiseconds (0-99).

Scope

`PUBLIC`

3.1.5 uint8_t GPS::get_position(Error *errors, double *lat, double *lon)

Arguments

- `errors` is a pointer to an `Error` object, all the errors that occur will be saved to this pointer
- `lat` is a pointer to a `double` where the latitude of the Can will be stored.
- `lon` is a pointer to a `double` where the longitude of the Can will be stored.

Returns

`uint8_t` - The amount of errors that occurred.

Description

This method will get the current location (latitude and longitude) of the Can and save it to the `lat` and `lon` pointers.

Scope

`PUBLIC`

3.1.6 uint8_t GPS::get_altitude(Error *errors, double *altitude)

Arguments

- `errors` is a pointer to an `Error` object, all the errors that occur will be saved to this pointer
- `altitude` is a pointer to a `double` where the altitude of the Can will be stored.

Returns

`uint8_t` - The amount of errors that occurred.

Description

This method will get the current altitude of the Can and save it to the `altitude` pointer.

Scope

`PUBLIC`

3.2 Can

The Can class contains all the methods and properties needed to do measurements and calculations.

3.2.1 void Can::Can(HardwareSerial *radioSerial, HardwareSerial *gpsSerial, int radioSetPin, uint8_t ticksPerSecond)

Parameters

- `radioSerial` is a pointer to a `HardwareSerial` object (e.g. `Serial1`) used to communicate with the APC220 radio module.
- `gpsSerial` is a pointer to a `HardwareSerial` object (e.g. `Serial2`) used to communicate with the GPS module.
- `radioSetPin` is an `int` which holds the pin where the SET pin of the APC220 radio module is connected to.
- `tickPerSecond` is an `uint8_t` which tells how many times per second the `Can::tick` method gets called.

Returns

`void`

Description

This is the constructor of the class. All the needed instances are created here (`Adafruit_BMP280`, `Adafruit_MPU6050` and `GPS`) and some variables are set (`Can::radioSerial`, `Can::radioSetPin` and `Can::ticksPerSecond`)

Scope

`Public`

3.2.2 uint8_t Can::tick(Error *errors)

Parameters

- `errors` is a pointer to an `Error` object, all the errors that occur will be saved to this pointer

Returns

`uint8_t` - The amount of errors that occurred.

Description

The tick method will gather all the data from the modules. The gathered data will then be transmitted to the ground station and saved to the SD card that is installed on board.

Scope

`Public`