ELECTRICAL & ELECTRONIC ENGINEERING
STELLENBOSCH UNIVERSITY

Design (E) 314
Technical Report

High-Altitude Balloon Data Logger

Author:
Daniel Wait

Student Number:
20887507

May 20, 2019

Plagiaatverklaring / Plagiarism Declaration

1. Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
Plagiarism is the use of ideas, material and other intellectual property of another's work and to present is as my own.

2. Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
I agree that plagiarism is a punishable offence because it constitutes theft.

3. Ek verstaan ook dat direkte vertalings plagiaat is.
I also understand that direct translations are plagiarism.

4. Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelikse aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.

5. Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

[illegible]	20887507
Handtekening / *Signature*	Studentenommer / *Student number*
DBW	20 May 2019
Voorletters en van / *Initials and surname*	Datum / *Date*

Abstract

This document details the design process which was followed to implement the data logger of a High Altitude Balloon (HAB) which would be used for weather monitoring. The system requirements are stated and elaborated upon. The following sections tackle the hardware and software design which resulted in a functional system. Finally, a few recommendations are mentioned.

Contents

List of Figures

List of Tables

List of Abbreviations

HAB High Altitude Balloon

LCD Liquid Crystal Display

PCB printed circuit board

SB student's board

1 Introduction

E-Design 314 students were instructed to design and implement a data logger for a HAB using predetermined hardware. The student's board (SB) consists of an especially designed printed circuit board (PCB), the required component soldered to the baseboard, and a ST microcontroller for software processes.

The SB needs to be able to communicate with other devices, monitor its power consumption, read external sensor input, display data on a screen, and trigger a release mechanism under given conditions.

2 System description

The system can be separated into separate subsystems and processes based on the function of each component. The diagram below describes the interfaces between the processes which is required for the system to operate as desired.

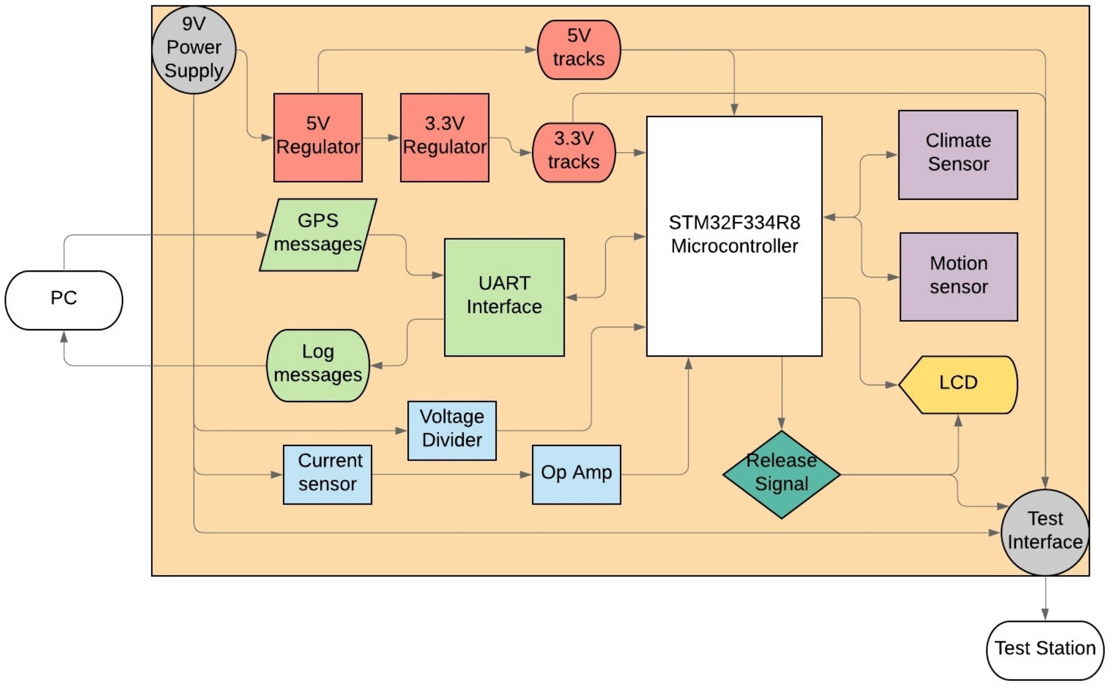


Figure 1: Block diagram of system operation

2.1 System Block Diagram

2.1.1 Power supply

A power supply provides a voltage which is converted to 5.0 and 3.3 by voltage regulators. The regulators' outputs connect to PCB tracks which power the STM32 board as well as various components on the SB.

2.1.2 UART communication

The SB is connected to a PC via a USB cable. The PC sends GPS messages in NMEA format which are converted to UART standard by the USB-UART interface and sent to the STM32 micro-controller. From the messages, the position and time data are extracted. Specifically, valid GPGGA messages are processed for this data. Each second, the micro-controller returns a log message which contains data from the various peripherals and sensors discussed in the following sections.

2.1.3 Current and Voltage monitoring

The 9 V supply to the board is sampled by voltage and current sensing circuitry. The current sensing chip is connected to both nodes of a resistor which is in series with the power supply. It detects the current through the series resistor and outputs a proportional current through a load resistor; the voltage over the load resistor is input to an operational amplifier with a controlled gain. The voltage divider is simply two high impedance resistors in series. The output of each circuit is connected to ADC pins on the microcontroller for processing.

2.1.4 Climate and Motion sensors

The climate sensor measures ambient temperature, atmospheric pressure, and humidity. The motion sensor detects acceleration in three axes. These sensors are configured as slaves on the same I2C interface. The master (STM32) controls the communication and requests data from the sensors each second.

2.1.5 Release mechanism

A release mechanism is triggered when a 'burn signal' is activated. This signal is output if the SB is not within a predefined longitude range.

2.1.6 LCD

A Liquid Crystal Display (LCD) screen displays the altitude in meters, the release mechanism's burn signal status, and the temperature in degrees Celsius.

2.2 Components

There is a wide range of components which could be used for each of the processes described before. However, the components originally used for this project proved to be reliable and well documented for potential users.

The fundamental electrical components used for the system are given in Table 1. The circuit design used to implement these components correctly will be discussed in detail in section 3.

Table 1: Electrical components

Function	Component
5.0 V regulator	LM7805CT/TO-220
3.3 V regulator	MCP1700-3302E/TO
USB-UART Interface	FT230XS
Current Sensor	ZXCT1008
Op-Amp	MCP6001UT-I/OT
Climate sensor	BME280
Motion sensor	LIS2DH12
LCD	PC1601-F

3 Hardware design and implementation

This section will discuss the hardware design and the calculations which motivated these choices.

3.1 Power supply

The board is supplied by a 9 V power supply. On the baseboard, there are two voltage regulators which supply components on the board with 5 V or 3.3 V regulated power.

The schematic in Figure 2 shows the circuit used to convert the power supply voltage to a 5 V source [1, p.5].

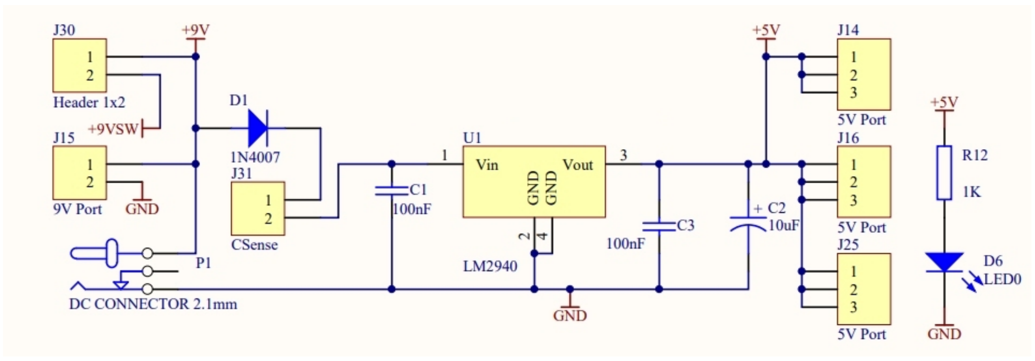


Figure 2: Power supply to 5V regulated output

3.1.1 Diode

A 1N4007 diode (D1) is placed directly after the power supply connection. This diode's function is to reject any negative voltage from the input. Thus, the rest of the components on the baseboard are protected against any damages a negative voltage may incur.

3.1.2 5 V regulator

The LM7805CT regulator can convert an input voltage ($\mathrm{V_{in}} = 7\,\mathrm{V}$ to $20\,\mathrm{V}$) to a fixed output voltage ($V_{\mathrm{out}} = 5\,\mathrm{V} typical \pm 0.25\,\mathrm{V}$) [2, p.3]. The STM32 board is externally powered by the 5V regulator and, internally, regulates the 5V to a 3.3V supply [1, p.4].

The capacitors on either side of the regulator serve to output a constant DC response. C1 is used when the regulator is an appreciable distance from the power supply [2, p.19]. Its purpose is to remove high-frequency components and noise. C2 and C3 improve the stability and transient response of the output response [2, p.19]. The capacitors store energy and, in the presence of voltage ripples, create a smoothing effect.

3.1.3 Heat Sink

To determine the heat-sink requirements of the LM7805CT, the operating temperature was compared to the thermal resistance multiplied by the maximum power consumption of the component.

$T_{OPR} = -40°C$ to $125°C$ [2, p.2]

$I_{LM(avg)} = 80\,\mathrm{mA}$

$V_{LM(max)} = V_{in(max)} - V_{out(min)} = 9.9\,\mathrm{V} - 4.75\,\mathrm{V} = 5.15\,\mathrm{V}$

$P_{LM(max)} = V_{LM(max)} \times I_{LM(avg)} = 412mW$

$\theta_{total} = \theta_{jc} + \theta_{ja} = 5°C/W + 65°C/W = 70°C/W$ [2, p.2]

$T_{max} = \theta_{total} \times P_{LM(max)} = 28.84°C$

The maximum temperature without a heat-sink is within the operating range of temperatures. Therefore, a heat-sink is not required.

3.1.4 Visual feedback

A 1 kΩ resistor (R12) in series with a red LED (D6) are connected to the 5V regulator output. This way, the user can visually see that the regulator is supplying a voltage.

3.1.5 3.3V regulator

The schematic in Figure 3 shows the circuit used to convert the 5V from the LM7805CT into a 3.3 V source.

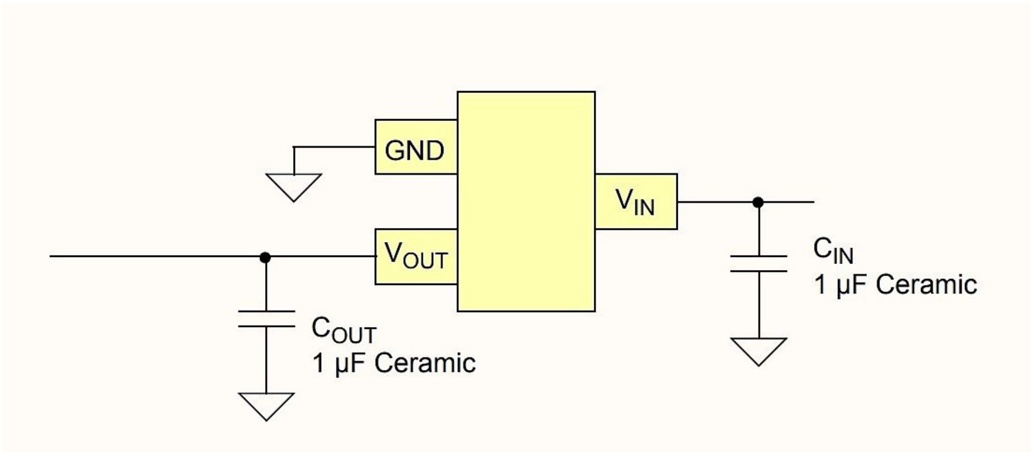


Figure 3: 3.3V Regulator circuit

The MCP1700 voltage regulator was used in a configuration suggested by the datasheet [3, p.2]. The capacitors, C_{in} and C_{out}, have the same purpose as the capacitors described for the LM7805CT circuit in section 3.1.2.

3.2 UART Communications

The USB-UART interface is controlled by a FT230XS integrated circuit. The circuit in Figure 4 is an adapted form of the USB Bus Powered Configuration given in the FT230XS datasheet [4, p.22]. The schematic was sourced from the 2019 Project Definition Document [1, p.6]. The series resistors

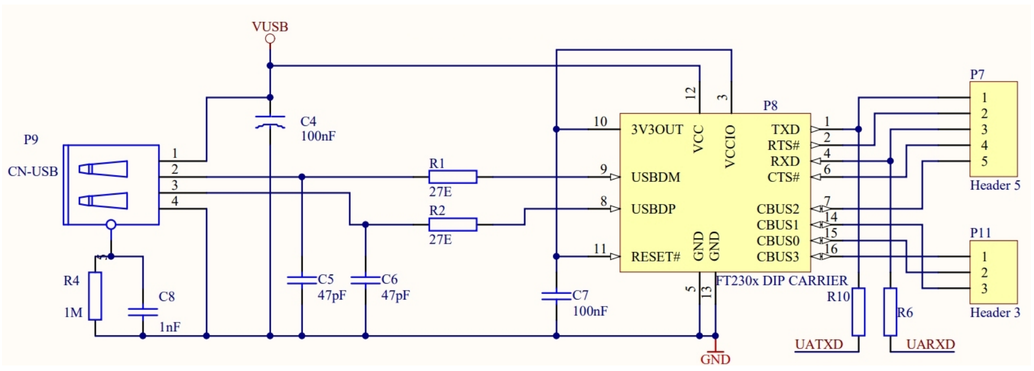


Figure 4: USB-UART Interface

on the TX and RX lines to the STM32 UART pins act as current limiters. They protect the pins on each device from drawing current higher than they can handle.

FT230XS:	$I_{IO(max)}$ =22 mA [4, p.16]
STM32F334R8:	$I_{IO(max)}$ = 25 mA [5, p.49]

$R_{6/10(min)} = \frac{3.3\,\text{V}}{22\,\text{mA}} = 150\,\Omega$

Therefore, 180 Ω resistors were used for R6 and R10.

3.3 LCD

The schematic in Figure 5 was sourced from the 2019 Project Definition Document [1, p.6].

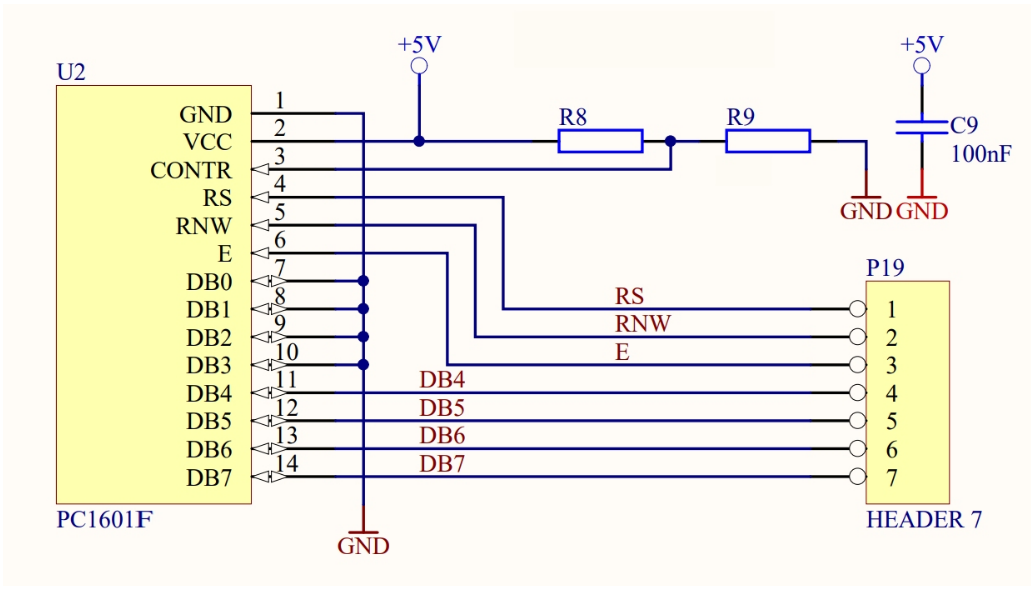


Figure 5: LCD connections

The LCD module used is a PowerTip PC1601-F. The LCD will be set up in 2-line mode to display the data from the logger. Pins RS (register select), RNW (read/not-write), E (enable), and DB7-4 (data bus lines) are connected to GPIO pins on the STM32 board. The LCD is powered with 5V, therefore it is best to use 5V tolerant FT/FTf pins [5, p.11]. Pins DB0-3 are at ground potential.

The capacitor (C9) has a smoothing effect on the 5V supply to the LCD. The resistors divide the input voltage to obtain an optimal contrast on the display. The recommended resistor values suggested in class resulted in a satisfactory contrast.

R8 = 22 kΩ and R9 = 680 Ω

3.4 Release Mechanism Signal

The signal uses TTL levels to represent the active\inactive states [1, p.2]. GPIO pins use TTa logic levels compliant with TTL levels [5, p.75]. Therefore, on the STM32 board, a GPIO pin acts as the 'burn signal' to a release mechanism when set to a logical high. It is connected to the Test Interface Connector via a wire.

3.5 STM32 I2C Pins

The STM32 was enabled for I2C communication. At J6, 10 kΩ pull-up resistors are connected between 3.3V and the SCL and SDA lines. SCL = PB6 and SDA = PB7.

3.6 Current measurement

Figure 6 is an adapted schematic from the 2019 Project Definition Document [1, p.9] which represents the circuit implemented to obtain the desired results.

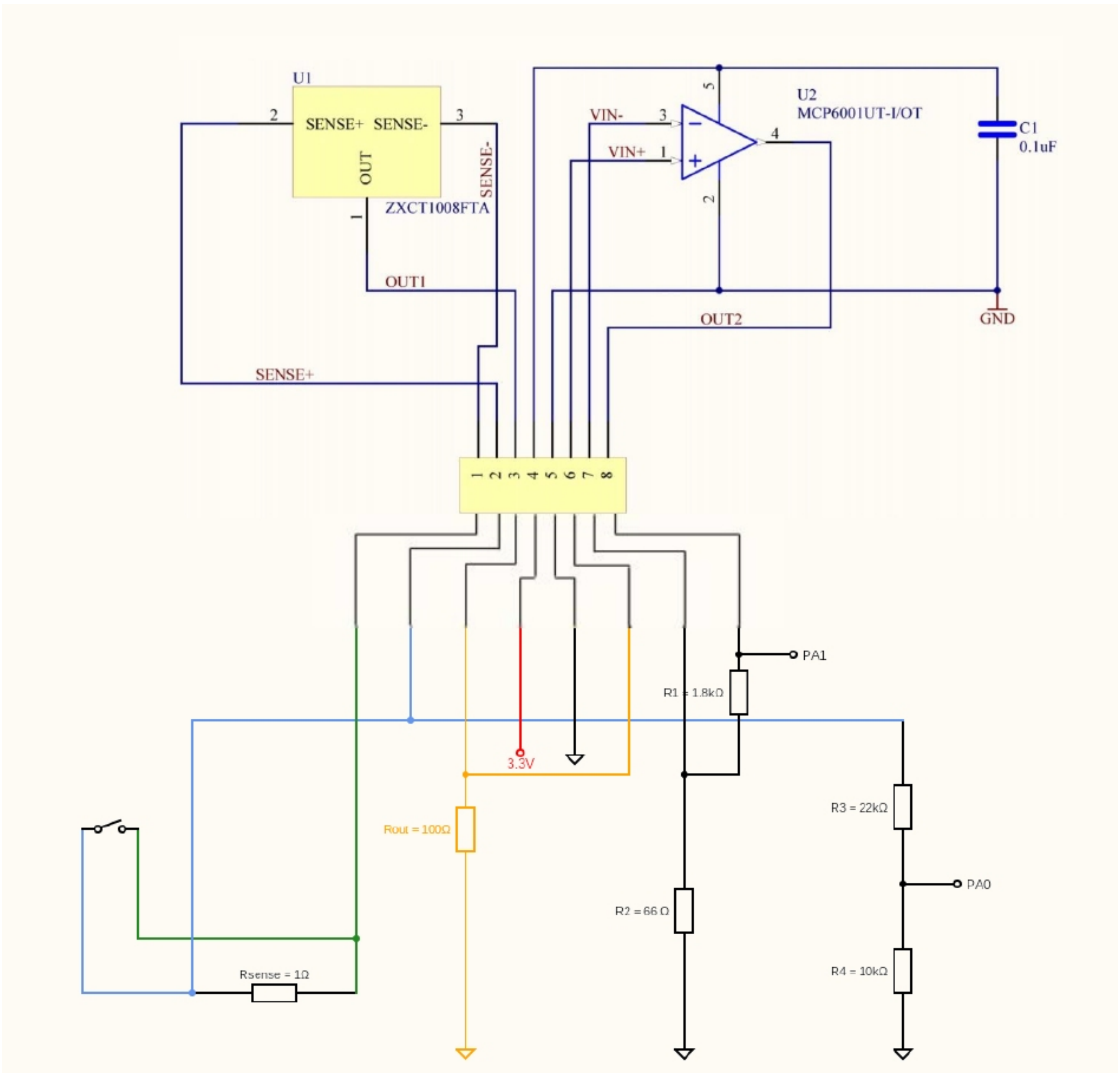


Figure 6: Voltage and Current sensor circuit

A breakout board with a current sensing integrated circuit (ZXCT1008FTA) and operational amplifier (MCP6001UT-I/OT) is soldered into J7 on the baseboard as well as resistors designed to monitor the current and voltage via ADC pins on the STM32 board.

The maximum voltage supplied to the board is 9.9V. However, maximum rated value of the STM32 I/O pin is 3.3V [5, p.35]. The series resistors connected between the positive terminal of J31 and ground act as a simple voltage divider. High impedance resistors were chosen so that the circuit would draw little current. To divide the input voltage by at least 3, $R_3 = 22\,\text{k}\Omega$ and $R_4 = 10\,\text{k}\Omega$.

A $1\,\Omega$/1W resistor (R_{sense}) is connected between the nodes of J31 - the current passing through this resistor is measured. The current sensor outputs a current through the load resistor ($R_{out} = 100\,\Omega$) proportional to the voltage across the terminals of R_{sense} according to the equation $V_{out} = 0.01 \times V_{sense} \times R_{out}$ [6, p.4]. Therefore the final equation of the current sensor becomes $V_{out} = I_{sense}$.

V_{out} is connected to the non-inverting input of the op-amp. The op-amp rails are connected to the 3.3V regulated source and ground. Thus the output to the ADC is limited to 3.3V, which prevents the op-amp output from exceeding the maximum rated value of the STM32 I/O pin. At the time of this design $I_{sense(max)} = 80\,\text{mA}$. The closed-loop controlled gain of the op-amp was designed such that, for $I_{sense} = 120\,\text{mA}$, the voltage output to PA1 is the maximum (3.3V).

$A_v = \frac{V_{max}}{I_{sense}} = \frac{3.3\,\text{V}}{120\,\text{mA}} = 27.5$

$A_v = 27.5 = 1 + \frac{R_1}{R_2}$

$\frac{R_1}{R_2} = 26.5$

Therefore, with standard resistor values $R_1 = 1.8\,\text{k}\Omega$ and $R_2 = 68\,\Omega$ the gain is closely matched by $A_v = 27.47$

3.7 Temperature, Humidity and Pressure Measurement

Figure 7 is an adapted schematic from the 2019 Project Definition Document [1, p.11] which represents the connections implemented on the baseboard. Red wire=3.3V; black wire=ground; yellow wire=PB6; green wire=PB7.

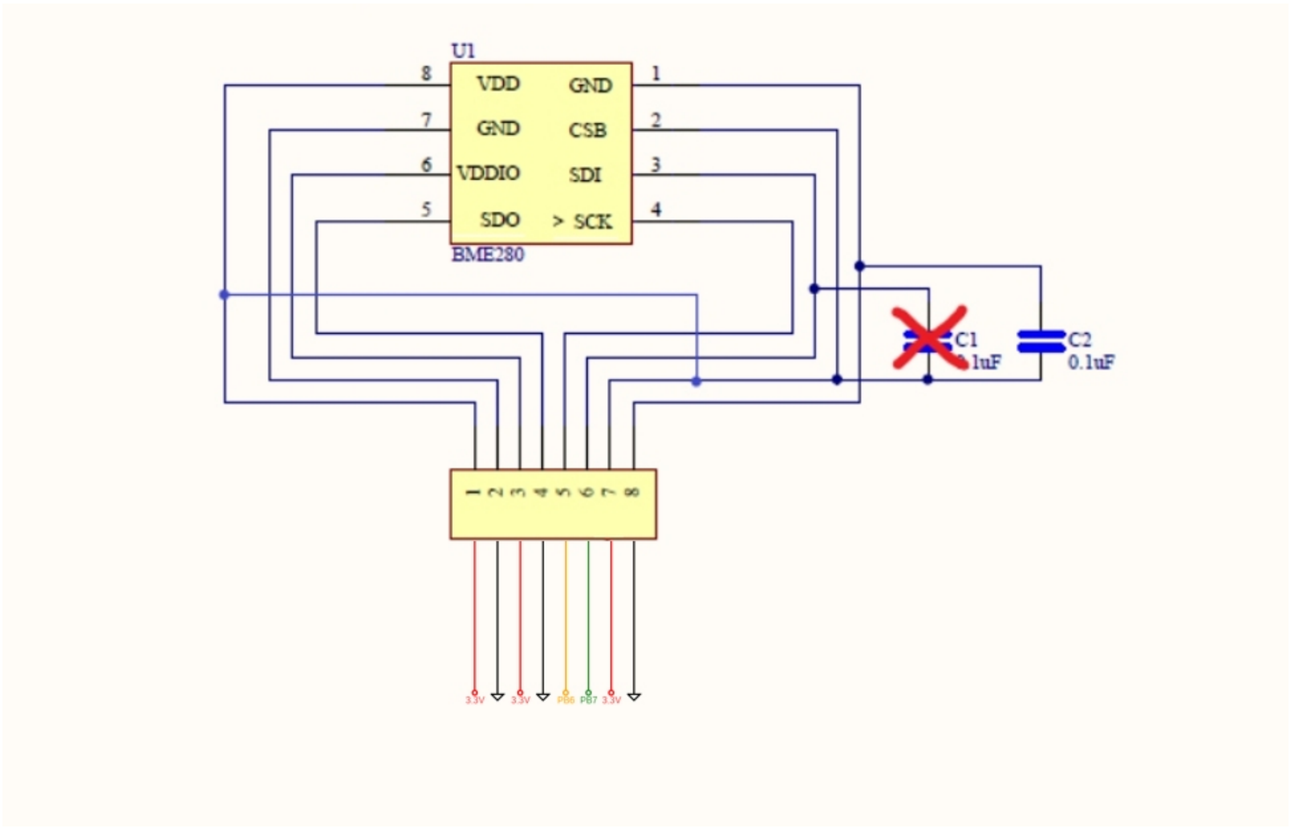


Figure 7: Climate sensor circuit

The Bosch Sensortec BME280 climate sensor is used for the temperature, humidity, and pressure. The BME280 is located on a breakout board soldered into the baseboard at J6. Due to complications in the manufacture of the breakout boards, there was no choice but to use the sensor in I2C configuration. owing to the numerous capacitors used in previous circuits, the loss of C1 will have no adverse effects on the sensor. The BME280 datasheet was consulted to decide on the specific connections indicated in Figure 7 [7, p.39]. It was decided to ground SDO, thus making the slave address of the sensor 0x76 [7, p.32]. The SCK and SDI pins of the BME280 are connected to the relevant I2C pins on the STM32 board - PB6 and PB7 respectively.

3.8 Linear Accelerometer Measurement

Figure 8 is an adapted schematic from the 2019 Project Definition Document [1, p.12] which represents the connections implemented on the baseboard. Red wire=3.3V; black wire=ground; yellow wire=PB6; green wire=PB7.

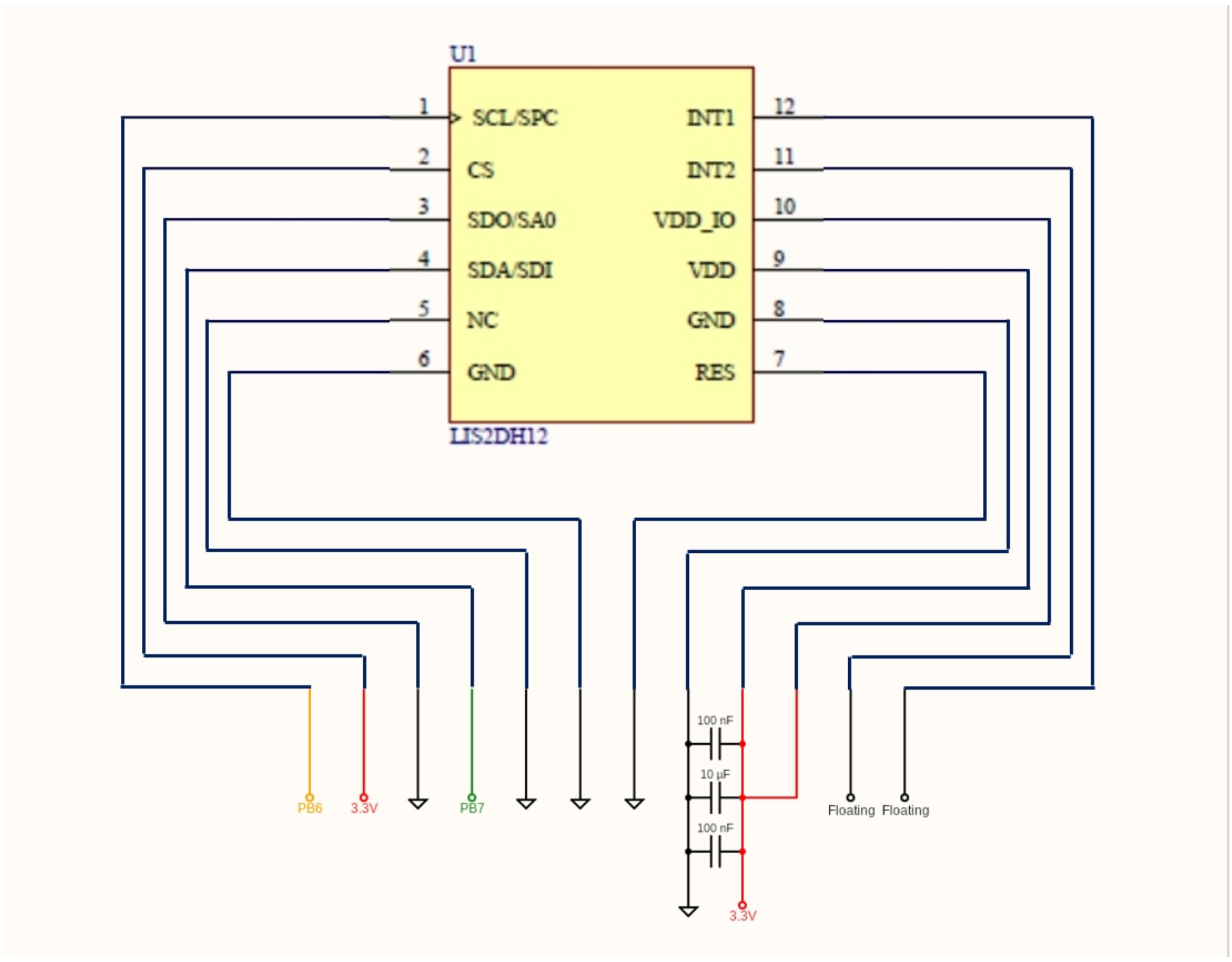


Figure 8: Motion sensor circuit

The ST LIS2DH12 is used for its accelerometer measurements. The circuit was built in J6 where the breakout board is oriented perpendicularly to the baseboard. The comparison of the axes of the baseboard versus breakout board are indicated in Table 2. The LIS2DH12 datasheet was consulted to decide on the specific connections indicated in Figure 8 [8, p.20]. Although the coupling capacitors were optional, I decided to implement the recommended circuit to be safe.

The decision to implement the accelerometer in I2C mode was purely tactical. Implementing the device in SPI mode may have come with the cost of time and functionality. Therefore, SCL/SPC and SDA/SDI were connected to PB6 and PB7 respectively. SDO/SA0 was grounded for convenience.

Table 2: Equivalent axes

Board Axis	LIS2DH12
x	-z
y	y
z	-x

4 Software design and implementation

The software implemented was designed with the goal of creating an efficient data-logger within the parameters specified by the 2019 Project Definition Document and Demo Guides.

The program cycle is fairly linear with the exception of the timer and UART interrupts. The interrupt callbacks set flags which are integrated into the user functions. The system operates with the purpose of receiving NMEA GPS strings, fetching data from the sensors, displaying data on the LCD, and sending a log message containing relevant data.

4.1 Flow diagram

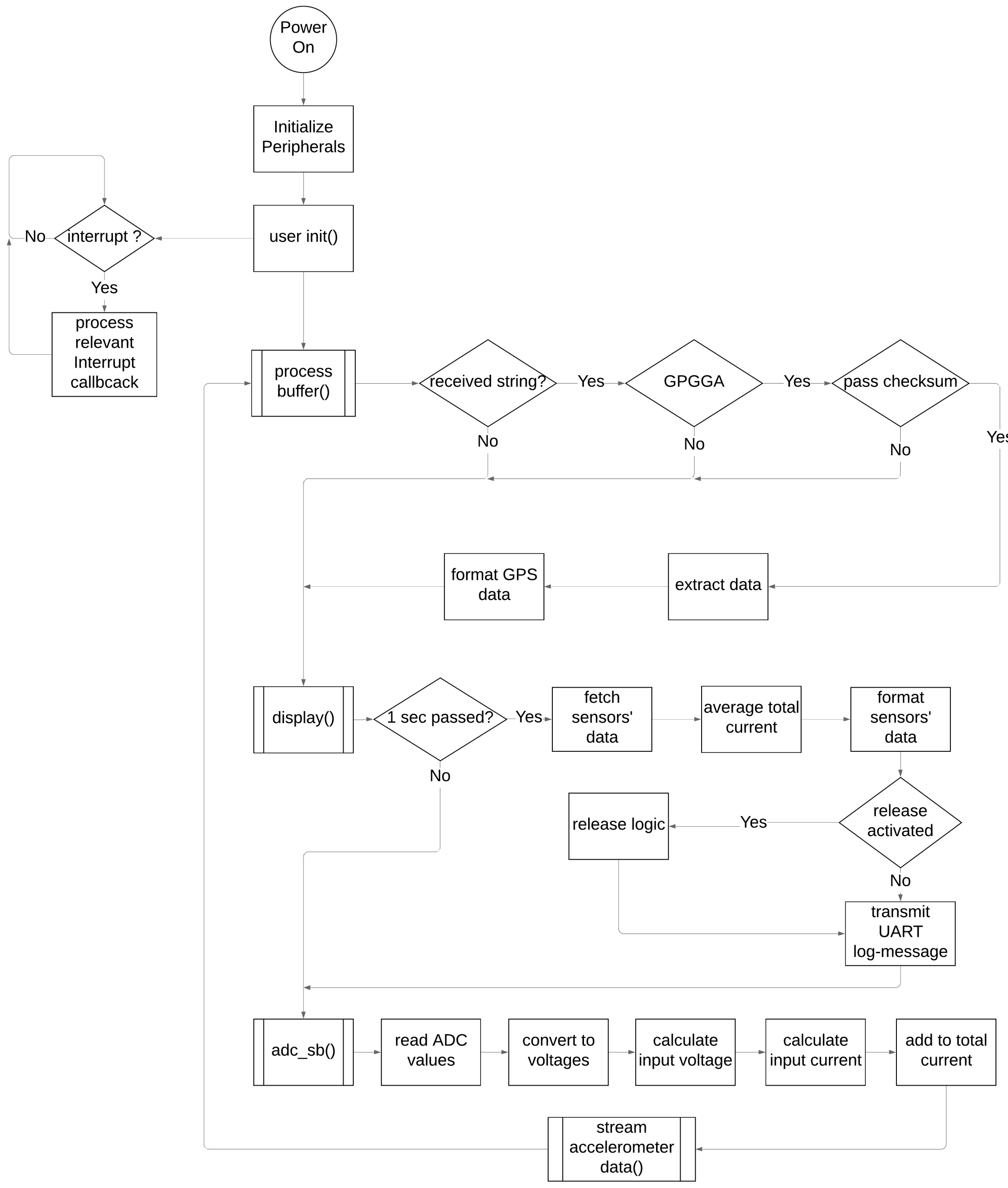


Figure 9: Program flow diagram

4.2 User functions

Herewith is a description of the user-defined functions and adapted functions which were essential for the required operation of the program. Most subordinate functions which were called within these functions will not be discussed directly.

4.2.1 void user_init(void)

This function is called in 'main.c' before the infinite while loop to initialize peripherals, sensors, and the LCD in the desired mode.

So that GPS messages can be received, the UART buffer is primed to receive data in interrupt mode. An interrupt timer with a one second period is started for the log messages. The climate sensor is enabled with the settings recommended for weather monitoring. The motion sensor is enabled, but its built-in temperature sensor is disabled. Lastly, the LCD is initialized for 2-line display, 4-bit operation, and the incremental, blinking cursor visible.

4.2.2 void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)

For each character received via UART, the callback adds the character to a buffer array (rx_buffer). This function populates the buffer array based on the standard format of a NMEA GPS string ('$xxx...\n'). Once the end of the string is detected, a flag is set to indicate that the buffer may be read. The flag is reset when the buffer is processed.

4.2.3 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)

When the one second period of the interrupt timer has elapsed, a flag is set to indicate that the next log message should be sent. The flag is reset when the message is sent.

4.2.4 void process_buffer(void)

This function is called repeatedly by the infinite while loop in 'main.c'.
The UART buffer arrray (rx_buffer) is subject to tests. The GPS message needs to be in GPGGA format and it has to pass a checksum. Messages which do not pass these tests are ignored.

Data is extracted from the relevant fields of valid messages. The fields are comma delimited. Blank fields are ignored so that the previous values for longitude, latitude, altitude, and time are retained. Finally, the data is formatted to be output correctly in the log message.

4.2.5 void adc_sb(void)

This function is called repeatedly by the infinite while loop in 'main.c'.
The integer values from the ADC are fetched, the ADC values are converted to equivalent the voltages on pins PA0 and PA1. PA0 is the output from the voltage divider and PA1 is the output from the current sensor circuit. The equivalent voltages are then used to calculate the current and voltage which was input to the circuits.
The number of times the functions has been called is tracked and the current is summed on each call of the function. When the periodic log message is called, the sum of the currents is averaged. The number of samples and total current is reset to zero for the next period.

4.2.6 void lcd_data_write(uint8_t data)

This function writes data in two consecutive nibbles to the LCD. It is used frequently to write characters to the cursor position on the display.

4.2.7 void lcd_instr_write(uint8_t dbs)

This function writes instructions in two consecutive nibbles to the LCD. It is used frequently to set the address of the cursor.

4.2.8 void stream_sensor_data_forced_mode(struct bme280_dev *dev)

Using the Bosch Sensortec API, the sensor is enabled with the desired settings for oversampling, filtering, and operation mode. The temperature, pressure, and humidity readings are then fetched and stored in a structure.

4.2.9 void stream_accelerometer_data(void)

This function is called repeatedly by the infinite while loop in 'main.c'.
Accelerometer data is fetched using the ST Microelectronis LIS2DH12 API. The code was sourced from the GitHub repository containing the API.

4.2.10 void standardize_accel(void)

The magnitude of the acceleration due to gravity was frequently larger than $1g$. Therefore, since the HAB should not have any external forces applied to it other than gravity, the vector was normalized. The normalized vector was multiplied by $1000mg$ in order to represent the gravitational acceleration in each axis.

4.2.11 void display(void)

This function is called repeatedly by the infinite while loop in 'main.c'.
If the flag from the interrupt timer is set: data from the GPS messages and sensors is formatted into a string for transmission via UART, the 'burn signal' logic is processed, data is displayed on the LCD, and the interrupt timer flag is reset.

4.3 Peripheral setup

This section discusses the motivation for enabling peripherals with the specific configurations used in the program.

4.3.1 GPIO

FT/FTf (5V tolerant) pins are required to write control instructions to the LCD. Thus, pins PA8-12, PB8, and PB9 were selected. These pins were set in output mode because input to the STM32 from these pins is undesirable.

Due to the burn signal using TTL logic levels, any pin may be used [5, p.75/76]. Therefore, PB1 was selected and configured in output mode.

4.3.2 USART

USART1 was selected arbitrarily. The UART communication frame was prescribed to be 8N1 and 115200 baud [1, p.3]. To allow manipulation of incoming characters via 'HAL_UART_RxCpltCallback', the NVIC interrupt was enabled.

4.3.3 ADC

ADC1 was selected to do multi-channel readings from the voltage and current sensors. 8-bit resolution was satisfactory to determine the voltage at the pins within 13 mV with the benefit of a fast response. Through trial and error, the sampling time with the most accurate results was found to be 181.5 cycles.

4.3.4 I2C

I2C1 was enabled and used with default settings without issue.

4.3.5 TIM

A timer was required to set a flag each second. Therefore, basic timer TIM6 was used for simplicity. The auto-reload register and prescaler values were designed for a 1Hz timer.

$CLK\_FREQ = 64MHz$ and $ARR = PSC = 8000$

TIM6_Freq $= \frac{CLK\_FREQ}{(ARR)(PSC)} = 1Hz$

4.4 Logic for release signal

In the program 'void burn_signal(void)' represents the logic required for the release mechanism. This function was sourced from 2019 E-Design 314 Test 2 and slightly modified.
The burn signal is activated for ten seconds, if, for five consecutive valid GPS messages:

altitude $>= 10000m$ and longitude not $\in \{17.976343, 18.9354\}$

The burn signal shall only be triggered once for the duration of the program.

5 Measurements and Results

The hardware and software design process took some trial and error in addition to trawling through datasheets for specific information. Confirmation that the final design is effective will be provided by comparisons and calculations.

5.1 Power supply

To check that the LM7805 voltage regulator outputs the desired voltage, two series resistors with a combined resistance of 48.9 Ω were connected as a parallel load to the regulated output pin of the LM7805. With a power supply voltage of 9.0 V, the voltage across the load was 5.07 V. Therefore, the current though the load would be approximately 104 mA. These results assert that the voltage is correctly regulated, and that the device is able to handle larger currents - which will be necessary as more components are added.

To confirm that the LM7805 operates as described in the datasheet, the power supply voltage was varied in 0.5 V decrements starting at 10 V - the maximum voltage supplied in the demos - until the voltage regulator was no longer operational. The results given in Table 3 below demonstrates that the LM7805 I used operates as desired according to the datasheet.

Table 3: LM7805 operation

Supply Voltage	Output Voltage
10 V - 7.5 V	5.08 V
7.0 V	5.06 V
6.5 V	4.52 V
6.0 V	0.00 V

5.2 UART communications

After soldering in the various elements and connectors required for the FT230XS module, I connected the TX and RX pins of the FT20XS (P7 pins 1 and 3) in order to do an echo-test from a PC using Termite software.

A variety of messages were parsed to the board using termite. The messages proved that the program can handle messages which are: non-GGA, invalid, or have blank fields.

After fulfilling the requirements up to Demo 4, I used the 'Demo Test App' provided on SUNLearn to check the consistency of the log messages from the board. The log message should be reported every second with a maximum drift of 100 m sec. The method to test the variance of the timing, was to enable certain processes which could influence timing accuracy and then actively change the input values. Table 4 below indicates the maximum and minimum variance in timing due to the described processes.

Table 4: Timing of log messages

Process	Minimum period	Maximum period
GPS emulating; varying input to all sensors/peripherals	970 m sec	1031 m sec
GPS emulating	987 m sec	1014 m sec
No inputs	997 m sec	1004 m sec

5.3 LCD

The hardware and software was well designed so that ASCII characters can be displayed on the screen with a legible contrast. Figure 10 depicts the output of the LCD with the burn signal enabled.

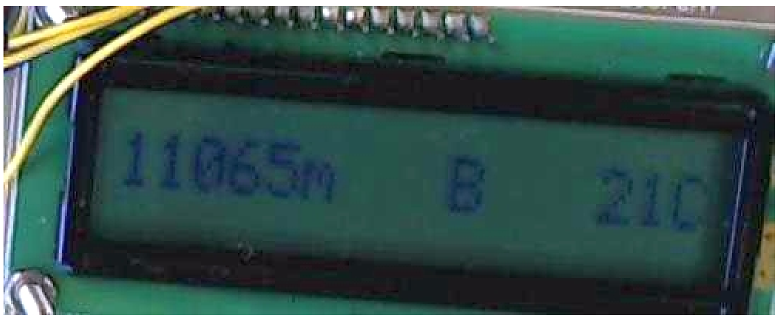


Figure 10: LCD output

5.4 Current measurement

The power supply was set to supply 9.9V to the board. A multimeter was used to measure the current delivered to the board. Due to the flashing LED on the STM32 board, the current alternated between 74.5 mA and 87.0 mA which, according to the designed gain, should translate to 2.05V and 2.39V output from the op-amp. However, the voltage on PA1 was measured to alternate between 2.1V and 2.45V. Therefore, the output from the circuit yields slightly larger voltages than expected.

The software converts the integer ADC values to a representation of the current. The calculated current fluctuated between 75 mA and 90 mA. This current measurement is satisfactorily similar to the real values mentioned above.

5.5 Pressure measurement

The software implementation of the BME280 API returned a pressure measurement of 100 kPa. Atmospheric pressure is 101.325 kPa on average. Therefore, the pressure is accurate within 2%.

5.6 Temperature measurement

The software implementation of the BME280 API returned a temperature measurement of $22.82°C$. According to a temperature sensor on my phone, the ambient temperature was $22°C$. Therefore, I assumed that the temperature sensor was functioning correctly.

5.7 Humidity measurement

The software implementation of the BME280 API returned a humidity measurement of 54%. According to the local weather statistics, the humidity was 47%. The difference in values can be attributed to the location of my measurement being the air-conditioned labs. The value is still considered reasonable.

5.8 Acceleration measurement

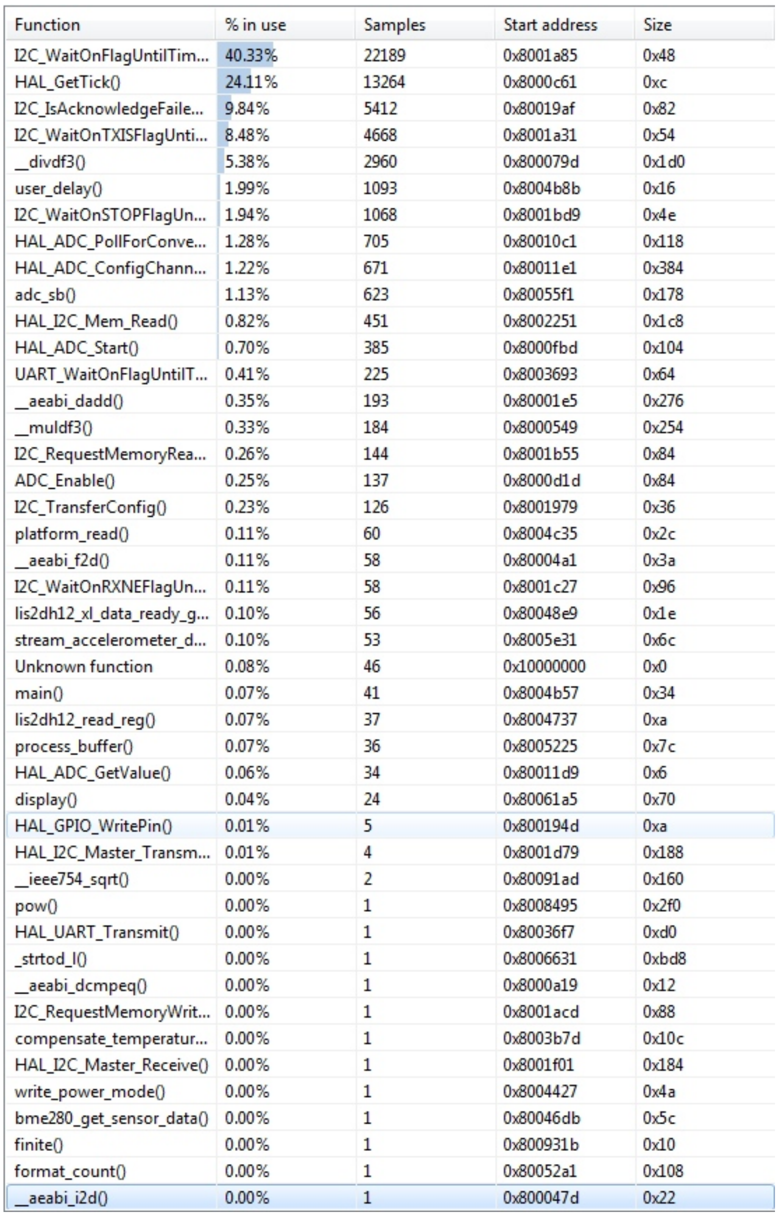
The axes were configured correctly so that tilting the board resulted in increased acceleration in the direction of the tilt.

5.9 Release signal

An oscilloscope was connected to PB1 to confirm that the burn signal was being activated when expected. The signal was activated as expected.

5.10 Software program (code profiling)

Atollic's SWV Trace Log was used to get an estimate of the percentage of time that each function was used. From the results in Figure 11, it is clear that the user defined functions are not demanding. Therefore, the software can be assumed to have an efficient design.



Function	% in use	Samples	Start address	Size
I2C_WaitOnFlagUntilTim...	40.33%	22189	0x8001a85	0x48
HAL_GetTick()	24.11%	13264	0x8000c61	0xc
I2C_IsAcknowledgeFaile...	9.84%	5412	0x80019af	0x82
I2C_WaitOnTXISFlagUnti...	8.48%	4668	0x8001a31	0x54
__divdf3()	5.38%	2960	0x800079d	0x1d0
user_delay()	1.99%	1093	0x8004b8b	0x16
I2C_WaitOnSTOPFlagUn...	1.94%	1068	0x8001bd9	0x4e
HAL_ADC_PollForConve...	1.28%	705	0x80010c1	0x118
HAL_ADC_ConfigChann...	1.22%	671	0x80011e1	0x384
adc_sb()	1.13%	623	0x80055f1	0x178
HAL_I2C_Mem_Read()	0.82%	451	0x8002251	0x1c8
HAL_ADC_Start()	0.70%	385	0x8000fbd	0x104
UART_WaitOnFlagUntilT...	0.41%	225	0x8003693	0x64
__aeabi_dadd()	0.35%	193	0x80001e5	0x276
__muldf3()	0.33%	184	0x8000549	0x254
I2C_RequestMemoryRea...	0.26%	144	0x8001b55	0x84
ADC_Enable()	0.25%	137	0x8000d1d	0x84
I2C_TransferConfig()	0.23%	126	0x8001979	0x36
platform_read()	0.11%	60	0x8004c35	0x2c
__aeabi_f2d()	0.11%	58	0x80004a1	0x3a
I2C_WaitOnRXNEFlagUn...	0.11%	58	0x8001c27	0x96
lis2dh12_xl_data_ready_g...	0.10%	56	0x80048e9	0x1e
stream_accelerometer_d...	0.10%	53	0x8005e31	0x6c
Unknown function	0.08%	46	0x10000000	0x0
main()	0.07%	41	0x8004b57	0x34
lis2dh12_read_reg()	0.07%	37	0x8004737	0xa
process_buffer()	0.07%	36	0x8005225	0x7c
HAL_ADC_GetValue()	0.06%	34	0x80011d9	0x6
display()	0.04%	24	0x80061a5	0x70
HAL_GPIO_WritePin()	0.01%	5	0x800194d	0xa
HAL_I2C_Master_Transm...	0.01%	4	0x8001d79	0x188
__ieee754_sqrt()	0.00%	2	0x80091ad	0x160
pow()	0.00%	1	0x8008495	0x2f0
HAL_UART_Transmit()	0.00%	1	0x80036f7	0xd0
_strtod_l()	0.00%	1	0x8006631	0xbd8
__aeabi_dcmpeq()	0.00%	1	0x8000a19	0x12
I2C_RequestMemoryWrit...	0.00%	1	0x8001acd	0x88
compensate_temperatur...	0.00%	1	0x8003b7d	0x10c
HAL_I2C_Master_Receive()	0.00%	1	0x8001f01	0x184
write_power_mode()	0.00%	1	0x8004427	0x4a
bme280_get_sensor_data()	0.00%	1	0x80046db	0x5c
finite()	0.00%	1	0x800931b	0x10
format_count()	0.00%	1	0x80052a1	0x108
__aeabi_i2d()	0.00%	1	0x800047d	0x22

Figure 11: SWT Trace log statistics

6 Conclusions

The previous sections have discussed the design process in-depth. However, there is always room for improvement.

6.1 Non-compliance and shortcomings

The final Demo went very well and there were no obvious errors. Even so, I am quite certain that - given a more strenuous set of tests - my ADC current conversion would be non-compliant. The ADC conversion is marginally within tolerance despite the fact that I did not use any correction algorithms for the conversion.

The 'stream_accelerometer_data' function called in the infinite while loop is a byproduct of the final testing phase of the accelerometer with the full project. Although it makes little difference to the apparent functionality of the code, it is redundant as the data only needs to be fetched periodically.

In 'myfunctions.c' there are plenty of globally declared variables.Many of these variables are not used more than once. Therefore, it would be more efficient to declare these variables locally.

6.2 Improvements

It is evident from Figure 11 that the APIs used to fetch sensor data require plenty time and memory. Had time been less constrained, writing user-defined functions to get data from the sensors may have made the code vastly more compact and efficient.

References

[1] A. Barnard, L. Visagie, and J. Treurnicht, "Ontwerpe314-projek," 2019.

[2] *3-Terminal 1 A POSITIVE VOLTAGE REGULATORS*, LM78XX datasheet, Fairchild Semiconductor Corporation, 2014.

[3] *Low Quiescent Current LDO*, MCP1700 datasheet, Microchip Technology Inc., 2018.

[4] *FT230X USB to Basic UART IC*, FT230X Datasheet, Future Technology Devices International Ltd, 2016.

[5] *STM32F334x4 STM32F334x6 STM32F334x8*, STM32F334R8 datasheet, ST, 2017.

[6] *INCREASED AMBIENT TEMPERATURE HIGH-SIDE CURRENT MONITOR*, ZXCT1008 datasheet, Diodes Incoporated, 2017.

[7] *BST-BME280-DS002-15*, BME280 - Data sheet, Bosch, 2018.

[8] *MEMS digital output motion sensor*, LIS2DH12, ST, May 2017.

7 Appendix

Table 5: STM32 Pins Activated

Pin Location	Configuration
PA0	ADC1_IN1
PA1	ADC1_IN2
PA8	GPIO_Output
PA9	GPIO_Output
PA10	GPIO_Output
PA11	GPIO_Output
PA12	GPIO_Output
PB1	GPIO_Output
PB6	I2C1_SDA
PB7	I2C1_SCL
PB8	GPIO_Output
PB9	GPIO_Output
PC4	USART1_TX
PC5	USART1_RX