Final Project

This document addresses the Final Project Instructions for the Making Embedded Systems class.

The objective is to create Propane Tank Weight Sensor System with something considerably more accurate and precise to replace the stock mechanical spring gauge:

Minimum Project Requirements

The project must:

(a) Use a Cortex-M processor:

This project is using an STMicro STML4, specifically the STM32L475VG, part of the STM32 Ultra Low Power Arm Cortex-M4 32-bit MCU+FPU series found on the B-L475E-IOT01A Discovery Board.

The prototype uses the Discovery kit for IoT node.

(b) Have a button that causes an interrupt

This project leverages the code from Exercise 4 that implements an operational mode/state switch via interrupt-driven button press code. There's a small button library as well as the interrupt handler that deals with button presses and system state changes.

(c) Use at least three peripherals such as ADC, DAC, PWM LED, Smart LED, LCD, sensor, BLE

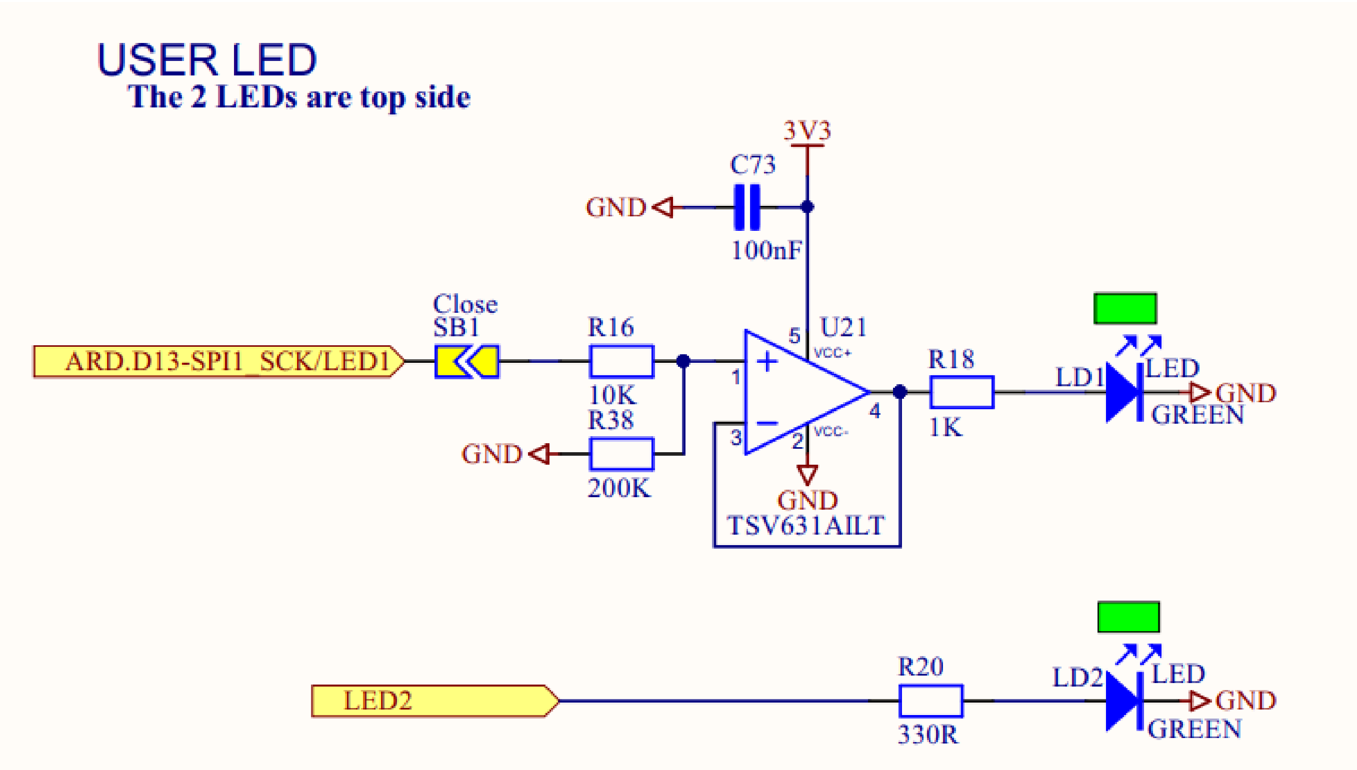
The peripherals used in the project:

External:

- Load cell (weight sensor) such as the SparkFun HX711. See HX711 code.
- Display: SSD1306 OLED Dual Color (Yellow / Blue) such as this I2C Serial 12864 on Amazon that includes mbedded Driver IC. See SSD1306 code
- Temperature Sensor (The onboard device TODO specify)

Internal

- Onboard LED: See LED code



- The built-in STM32L475 Flash is used to save tare weight offset. (see code). There's a check for button long press in the RTOS Display Thread, here. Upon detecting a button long press, the display thread is paused and a new system `Tare` state is assigned. Upon entering the Tare State the the display is cleared, the message "Tare" is displayed, and the calculated offset is saved to flash via SaveDeviceConfig() from the DoScaleTare() task.

See the customized linker file that placed Flash memory starting at `FDATA = 0x080FF000`

```
MEMORY
{
    FLASH (RX) : ORIGIN = 0x08000000, LENGTH = 996K
    FDATA (R)  : ORIGIN = 0x080FF000, LENGTH = 4K
    SRAM (RWX) : ORIGIN = 0x20000000, LENGTH = 96K
    RAM2 (RWX) : ORIGIN = 0x10000000, LENGTH = 32K
}
```

The Flash configuration is mapped to the static const struct FlashConfig FLASH_CONFIG in Flash memory, along with a copy in an updatable RAM cache value:

```
    // FLASH_CONFIG is the data actually on the flash. See DeviceFlashConfig()
    __attribute__((__section__(".flash_user_data"))) static const struct
FlashConfig FLASH_CONFIG;

    // CACHE_CONFIG is the runtime, updatable copy of the config. See
DeviceCacheConfig()
    static struct FlashConfig CACHE_CONFIG;
```

- The on-board LPS22HB barometric sensor was used in this project (see code). Currently the pressure reading is sent to the UART in the RTOS LED Thread #11. To make things interesting from a multi-threadced RTOS perspective, the pressure is also read in the experimental PWM Thread.

- PWM Timers and watchdogs

There are some experiments with STM32 PWM in the PWM Thread, but this was done only for educational purposes and will be removed from the final project.

See Page 44 of the STM32L475xx Datasheet (DS10969):

3.23 Timers and watchdogs

The STM32L475xx includes two advanced control timers, up to nine general-purpose timers, two basic timers, two low-power timers, two watchdog timers and a SysTick timer. The table below compares the features of the advanced control, general purpose and basic timers.

Table 11. Timer feature comparison

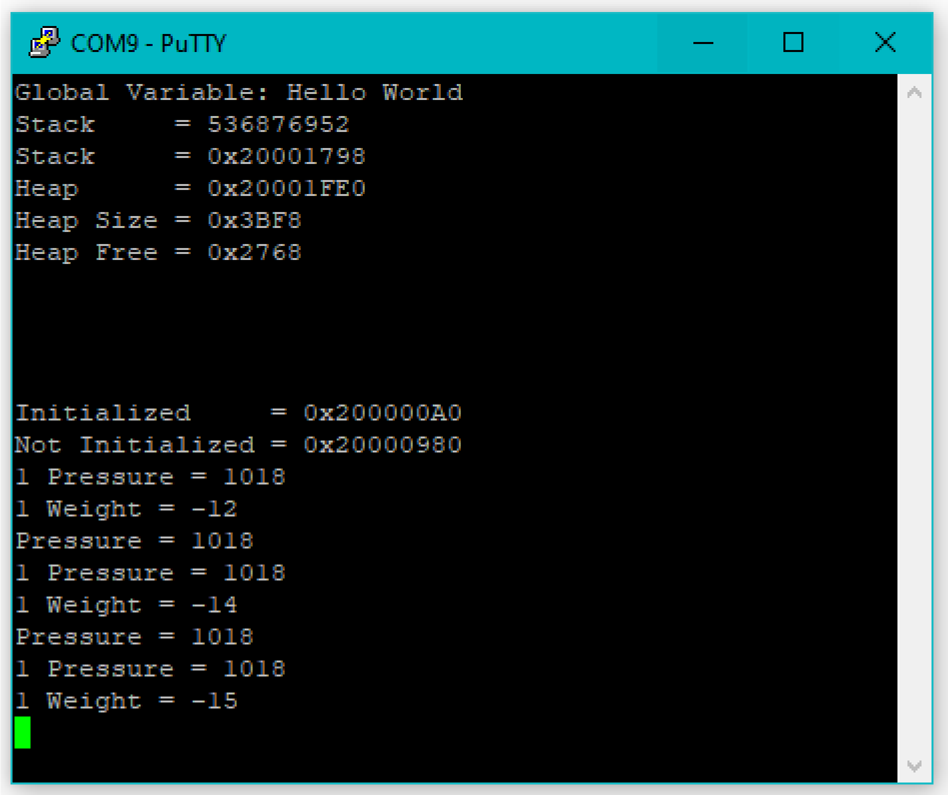
Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complementary outputs
Advanced control	TIM1, TIM8	16-bit	Up, down, Up/down	Any integer between 1 and 65536	Yes	4	3
General-purpose	TIM2, TIM5	32-bit	Up, down, Up/down	Any integer between 1 and 65536	Yes	4	No
General-purpose	TIM3, TIM4	16-bit	Up, down, Up/down	Any integer between 1 and 65536	Yes	4	No
General-purpose	TIM15	16-bit	Up	Any integer between 1 and 65536	Yes	2	1

(d) Have serial port output

See UART code. Several of the RTOS threads send data to the UART, including the main LED Thread1 and the experimental PWN Thread.

An RTOS-safe semaphore wrapper was created around the `HAL_UART_Transmit` found in the STMicroelectronics / stm32l4xx_hal_driver / stm32l4xx_hal_uart.c.

Sample UART output at startup:



(e) Implement an algorithmic piece that makes the system interesting

> *Every sensor is a temperature sensor. Some are better than others.* --Unknown / Elecia White

Consider measuring weight over time and temperature.

TODO: *interesting*

(f) Implement a state machine

There's currently a prototype LED State Machine with IsBlinking, AlwaysOn, AlwaysOff states.

Not required to use a HAL (but it is encouraged)

This project uses the STM32L4XX HAL, for example here, and is a multi-threaded application using embedded RTOS (specfically CMSIS_RTOS) for example included here.

Code that uses the STM32 HAL will need the `#include <stm32l4xx_hal.h>`. Future versions of this codebase should include a hardware conditional include such as this example:

```
#if defined(STM32F0)
#include "stm32f0xx_hal.h"
#elif defined(STM32F1)
#include "stm32f1xx_hal.h"
#elif defined(STM32F4)
#include "stm32f4xx_hal.h"
#include "stm32f4xx_hal_gpio.h"
#elif defined(STM32L0)
#include "stm32l0xx_hal.h"
#elif defined(STM32L1)
```

```
#include "stm32l1xx_hal.h"
#elif defined(STM32L4)
#include "stm32l4xx_hal.h"
#elif defined(STM32F3)
#include "stm32f3xx_hal.h"
#elif defined(STM32H7)
#include "stm32h7xx_hal.h"
#elif defined(STM32F7)
#include "stm32f7xx_hal.h"
#elif defined(STM32G0)
#include "stm32g0xx_hal.h"
#elif defined(STM32G4)
#include "stm32g4xx_hal.h"
#else
#error "Hardware platform not supported."
#endif
```

List of the tasks to complete for the project

- [x] Confirm operational display
- [x] Confirm operational weight sensor
- [x] Show weight value on display
- [ ] Update docs on new I2C port being used for SSD1306
- [x] Confirm serial port operation
- [x] Implement Serial Rx/Tx
- [x] Implement serial debug messages
- [ ] Sleep serial port when inactive
- [ ] Implement Sleep / Wake-up
- [ ] Determine field power source
- [x] Design enclosure
- [ ] Print enclosure
- [ ] Mount hardware in enclosure

Challenges

There are plenty of potential challenges. This project was originally started (and abandoned) 6 years ago, in part due to mechanical issues. It is hoped that modern 3D Printing flexibility will be able to help with the mechanical mounting aspects.

Technical implementation difficulties of a new hardware platform are always a concern (the original project was based on the ESP8266 and the LUA language).

Details on some of the challenges:

Minimal Example Code

There is currently relatively little STM32CubeL4 example code for the B-L475E-IOT01A discovery board chosen for this project. Porting code between architectures is beyond the scope of the class.

Furthermore, the HX711 weight sensor interface and SSD1306 display are not built-in on the development board.

Mistakes in Example Code

I submitted STM32CubeL4/issues/61 and PR #60 to fix a problem in an example where there Source Address was the same as the Destination. (not only uninteresting, but failed with error)

External sensors and peripherals

Certainly one of the benefits of having an evaluation board is having the connections "built-in" and sample code readily available. Unfortunately the Discovery Board used did not have a display and the I2C HX711 load cell was of course external, adding the additional challenge of finding and wiring up the I2C connections.

The better part of a day was spent trying to get the SSD1306 Display to work on `I2C1` that the documentation claims is on `PB8` and `PB9`, but it was later learned the STM32CubeIDE shows GPIO pins `PB6` and `PB7` instead:

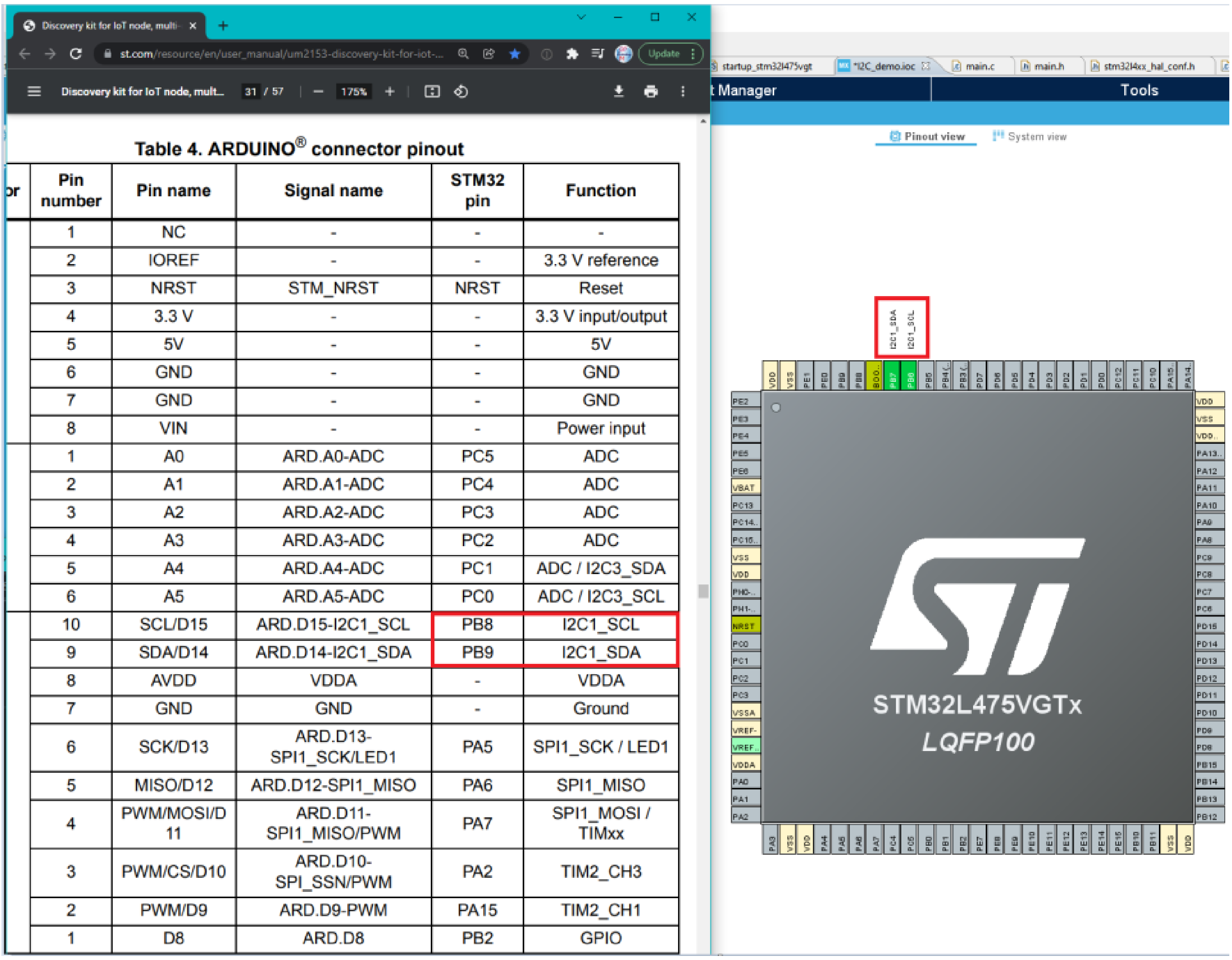
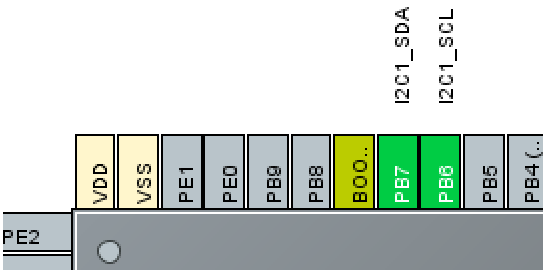


Table 4. ARDUINO® connector pinout

Pin number	Pin name	Signal name	STM32 pin	Function
1	NC	-	-	-
2	IOREF	-	-	3.3 V reference
3	NRST	STM_NRST	NRST	Reset
4	3.3 V	-	-	3.3 V input/output
5	5V	-	-	5V
6	GND	-	-	GND
7	GND	-	-	GND
8	VIN	-	-	Power input
1	A0	ARD.A0-ADC	PC5	ADC
2	A1	ARD.A1-ADC	PC4	ADC
3	A2	ARD.A2-ADC	PC3	ADC
4	A3	ARD.A3-ADC	PC2	ADC
5	A4	ARD.A4-ADC	PC1	ADC / I2C3_SDA
6	A5	ARD.A5-ADC	PC0	ADC / I2C3_SCL
10	SCL/D15	ARD.D15-I2C1_SCL	PB8	I2C1_SCL
9	SDA/D14	ARD.D14-I2C1_SDA	PB9	I2C1_SDA
8	AVDD	VDDA	-	VDDA
7	GND	GND	-	Ground
6	SCK/D13	ARD.D13-SPI1_SCK/LED1	PA5	SPI1_SCK / LED1
5	MISO/D12	ARD.D12-SPI1_MISO	PA6	SPI1_MISO
4	PWM/MOSI/D11	ARD.D11-SPI1_MISO/PWM	PA7	SPI1_MOSI / TIMxx
3	PWM/CS/D10	ARD.D10-SPI_SSN/PWM	PA2	TIM2_CH3
2	PWM/D9	ARD.D9-PWM	PA15	TIM2_CH1
1	D8	ARD.D8	PB2	GPIO

This closeup from the STM32CubeIDE clearly indicates `I2C1` GPIO pins are on `PB6` and `PB7`.



The SSD31306 configuration is currently using `I2C3` instead. (See also the SSD1306 default template)

Lesson learned: always do a simple IO level and control check on GPIO lines before starting somwthing more complex such as I2C communication.

Known library Problems

There's an open STMicroelectronics/stm32l0xx_hal_driver issue: An I2C NACK during memory address transfer goes undetected to be aware of that may impact I2C. This was stumbled upon when encountering a `I2C_WaitOnFlagUntilTimeout` problem, but that was related to the GPIO pin conflict, described above.

Mechanical support for weight sensor

I reached out on Twitter for suggestions on how to mount the load cell. One of the responses is regarding load cell damage if the propane tank is dropped into place.

Another concern is load cell creep when the tank is left in place for a long period.

There's no mention of "load creep" in the Avia Semicondictor HX711 Datasheet, but that seems like a legitimate concern. There is only one instance of the word "creep" on the Sparkfun Load Cell Amplifier HX711 Breakout Hookup Guide.

Yet another potential challenge pointed out on Twitter is the temperaturer drift over time of the load cell.

Physical Room

There's relatively limited room for the tank: so little, that there's a hole in the bottom of the cabinet to help with maneuvering when replacing a fresh tank:

Enclosure

Enclosures are always challenging. Fortunately there's a 3D Printer available to create a custom enclosure for this project.

The enclosure should probably be weather-proof, and located reasonably far from the grill heat box which can get up to 500 degrees.

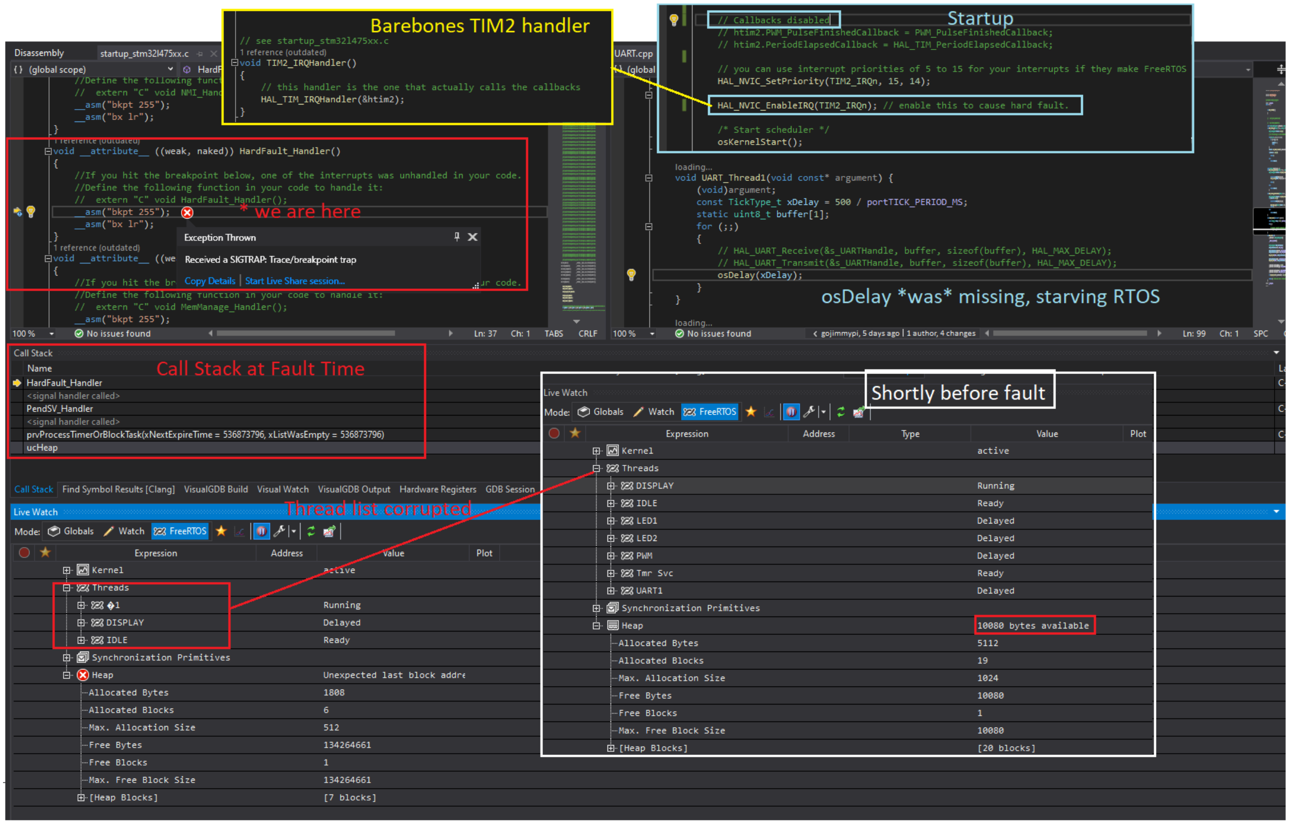
The OLED display is likely not tolerant to hard freeze.

Component Availability

If any sort of mass-production was desired, there's of course the chip shortage to be concerned about.

Hard Faults

Any list of challenges would not be complete without of course my Hard Faults!



Deliverables

The final project will be delivered as:

(a) Video of the system working as intended. See https://youtu.be/YloqKTbCUQQ

(b) Write up of the system (PDF or Google docs report). This document and Final Propject Report

(c) Link to the code: see GitHub IoT BBQ STM32 Project.

Instructions to build can be found in the solution directory README

Optional Bonus

Power analysis

Tips learned from Ben in class:

- When powering an embedded device from batteries, say a couple of AA cells, there's likely a voltage drop / fluctuation depending on what the processor is doing at any given moment, that may affect things like ADC.

See also the [PPK2 Power Analysis Setup Notes(https://gojimmypi.github.io/ppk2-power-analysis/).

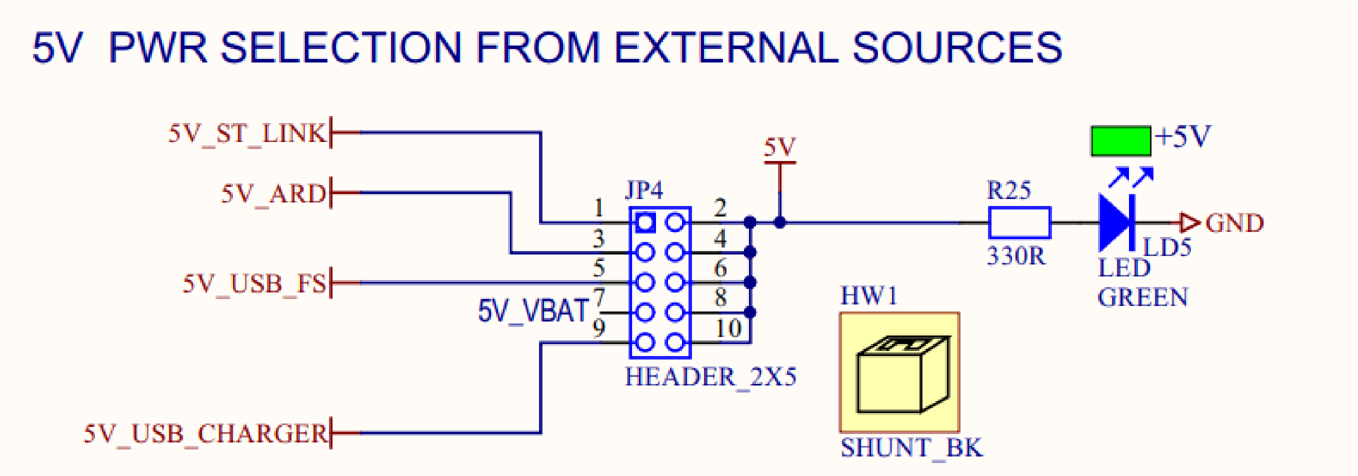
Firmware update

TODO

System profiling

TODO

The Power LED is always on, and needs to be unsoldered to not use it:



References

Mechanical

- Wikipedia Propane
- Statasys Grabcad Community Propane Tank

Core Hardware

- STM32 Discovery Kits
- STM32L4 series of ultra-low-power MCUs
- ARM MBED DISCO-L475VG-IOT01A (B-L475E-IOT01A (this is where the link on the OTG file points)
- Mouser 5 page B-L475E-IOT01A Data Brief
- Mouser Discovery Kit for IoT Node (this is the front and back of the insert card in for the shipped product)
- Mouser STMicroelectronics B-L475E-IOT01A Discovery Kit for IoT Node

Drivers

- ST-LINK, ST-LINK/V2, ST-LINK/V2-1, STLINK-V3 USB driver signed for Windows7, Windows8, Windows10

Peripheral Hardware

- SSD1306 I2C Serial
- Sparkfun Load Cell Amplifier HX711 Breakout Hookup Guide
- bogde/HX711
- nimaltd/HX711

Development Environment

- Microsoft Visual Studio 2019

- Sysprogs VisualGDB Extension
- CMSIS-RTOS2
- ST STM32L4 Discovery kit IoT node software
- ST STM32CubeMX STM32Cube initialization code generator
- ST STM32Cube MCU Package for STM32L4 series and STM32L4 Plus series (HAL, Low-Layer APIs and CMSIS, USB, TouchSensing, File system, RTOS)
- ST Description of STM32L4/L4+ HAL and low-layer drivers - UM1884

Tutorials and Sample Code

- Sysprogs VisualGDB Developing STM32 projects with Visual Studio tutorial
- Sysprogs VisualGDB Using the STM32 UART interface with HAL
- Sysprogs VisualGDB Using the I2C Interface on the STM32 Devices
- Sysprogs VisualGDB Creating Advanced STM32CubeMX-based Projects with VisualGDB
- Sysprogs VisualGDB Controlling STM32 Hardware Timers using HAL
- Sysprogs VisualGDB Using STM32 timers in PWM mode
- NordicPlayground mbed stm32f4xx_hal_uart
- afiskon/stm32-ssd1306 example; The code is included in this project.
- Sensors STM32CubeL4/Drivers/BSP/B-L475E-IOT01/
- github akospasztor/stm32-bootloader

Video Tutorials

- Digi-Key Getting Started with STM32 - Timers and Timer Interrupts by Shawn Hymel
- STMicroelectronics STM32CubeIDE basics - 05 TIM PWM HAL lab

Programming

- AVR035: Efficient C Coding for AVR

Coding Standards

- Wayback Machine GSFC C/C++ Coding Standards NASA Goddard Space Flight Center

Cloud Demo

- AWS AWS Cloud demonstration

Utilities

- VS Code Extension: Markdown PDF

Other Related Projects

- Honeybee Hive Monitoring - also uses HX711
- logicalelegance/midifun - sample project naming, directories, organziation.