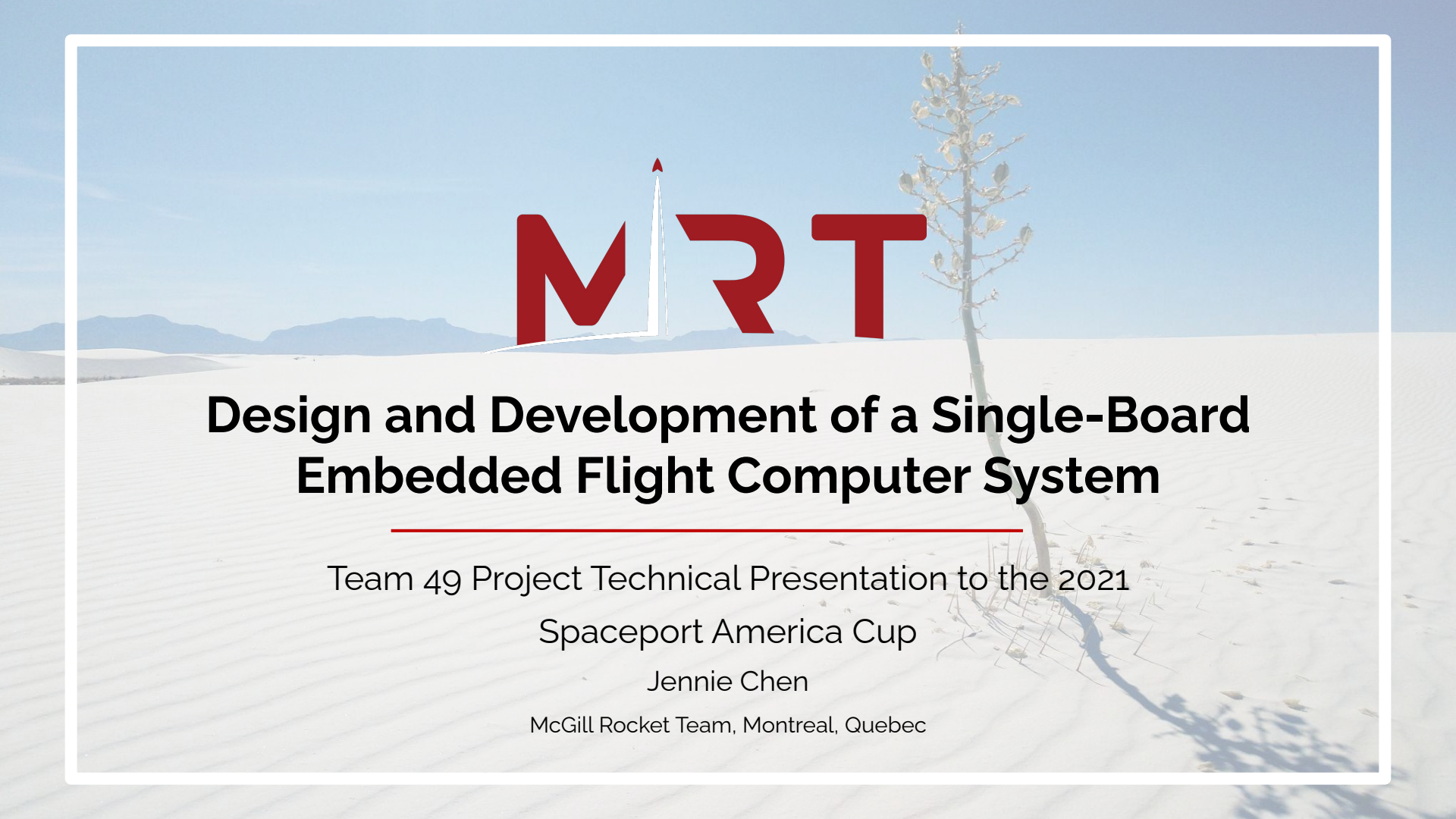


Design and Development of a Single-Board Embedded Flight Computer System

Team 49 Project Technical Presentation to the 2021 Spaceport America Cup

Jennie Chen

McGill Rocket Team, Montreal, Quebec

Agenda

The Team

Problem Definition

Development Process and Embedded Design

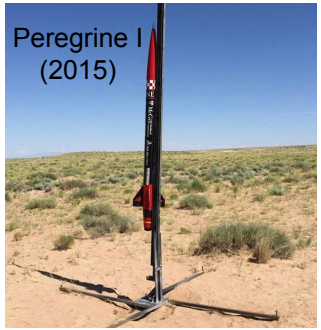
System Testing

Follow-on Work

Team History

- Founded in **2014**
- Participated in **5** IREC competitions
- Built **8** rockets over **5** years
- **1st** place overall at SAC 2018

Past Competition Launches



Problem Definition

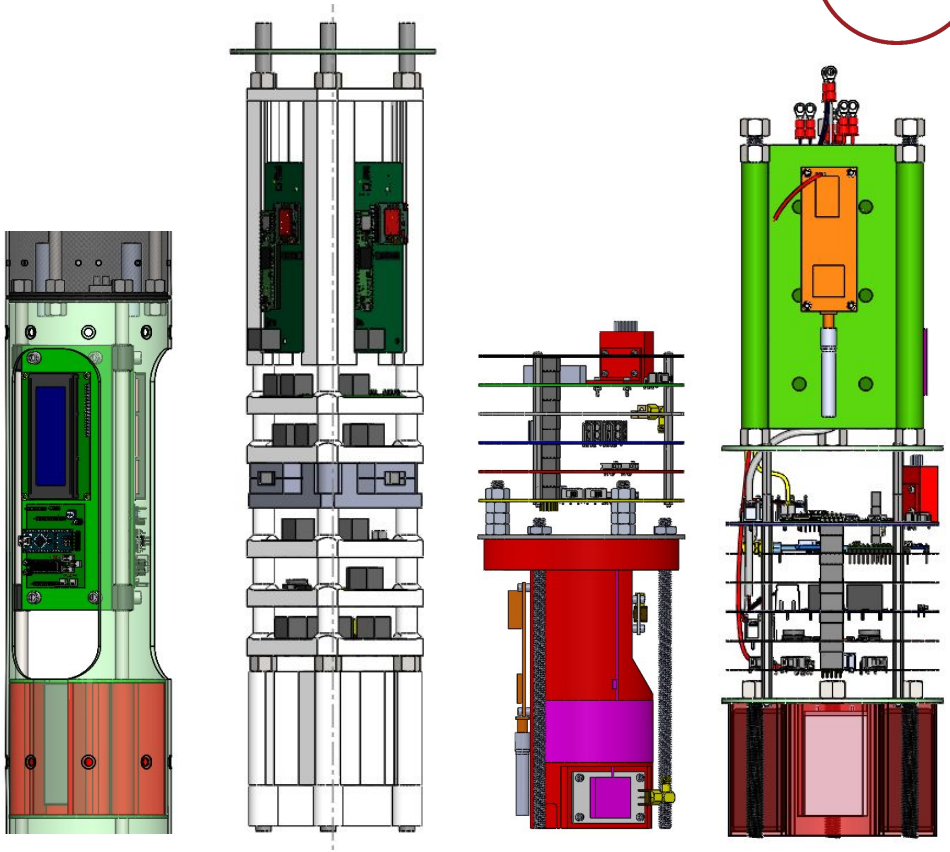
Avionics Bay

Modular design - limitations

- Space/weight **inefficient**
- Higher manufacturing **cost**

Motivation

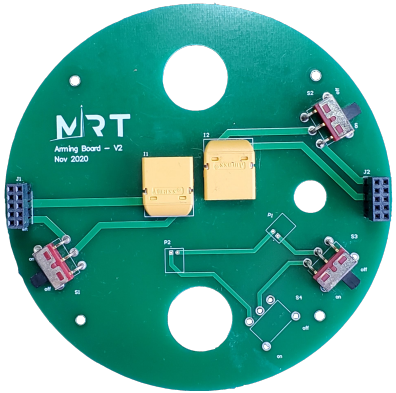
- Improve efficiency and increase redundancy
- Lighter avionics systems for 10k hybrid project

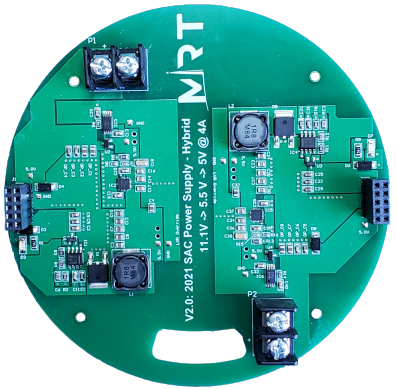


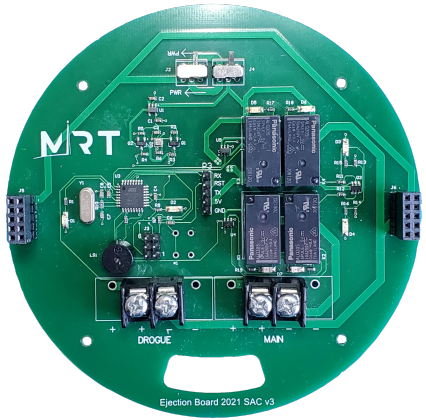
Past MRT avionics bay designs

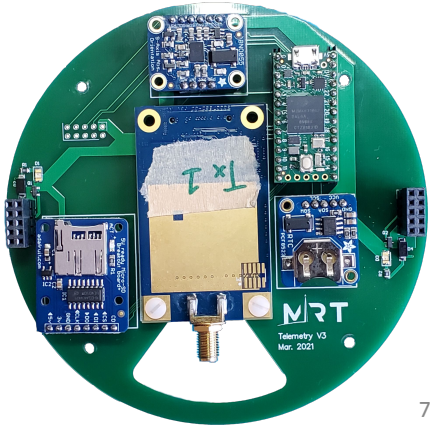
Current Avionics Systems (2021)

- Arming board
- Power management
- Ejection
- Telemetry
- Video recorder





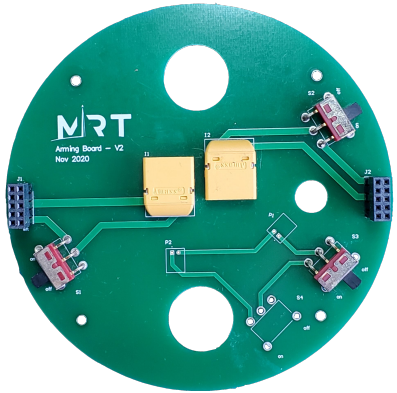


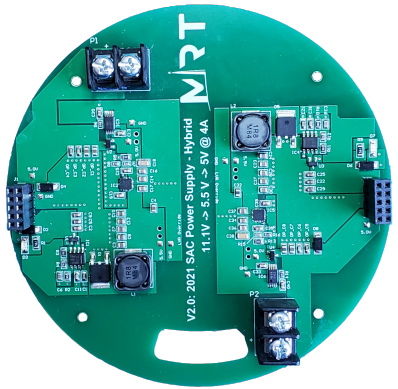


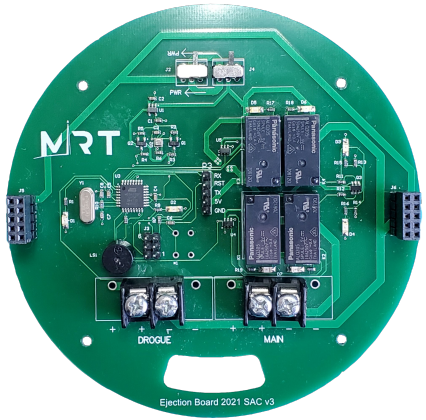
Current AV Bay PCBs

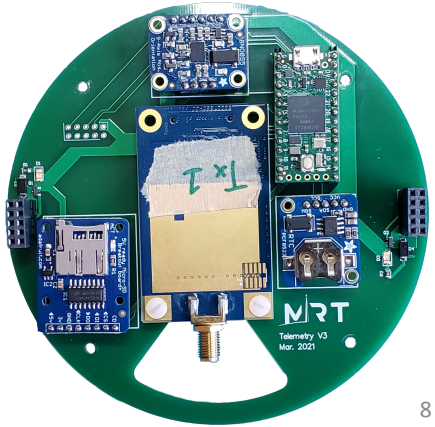
Current Avionics Systems (2021)

- **Arming board**
- **Power management**
- **Ejection**
- **Telemetry**
- Video recorder





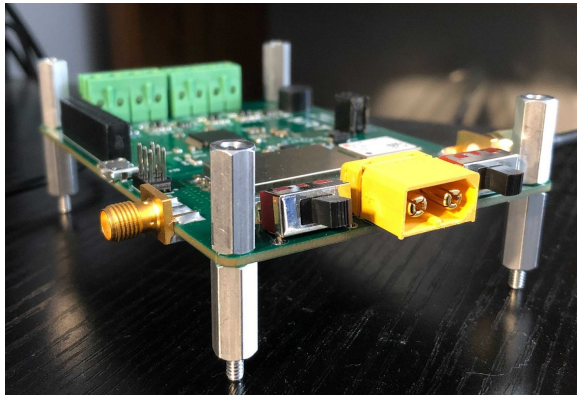




Current AV Bay PCBs

Solution

- Single-board flight computer
- **Merge** avionics systems
  - Power management
  - Ejection
  - Telemetry
  - Radio
- Set Requirements

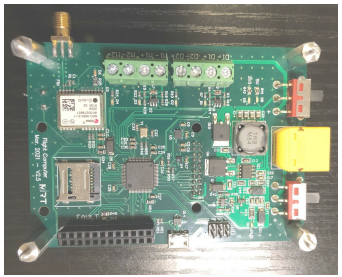


Assembled Flight Computer

HARDWARE	SOFTWARE
➔ SRAD radio ➔ SMD sensors and components ➔ Removable debug shield ➔ Solid-state ejection circuit	➔ STM32-based embedded solution ➔ Custom software drivers and features ➔ Real-time operating system

Development Process and Embedded Design

Process Overview



5: INTEGRATION

Integrating the component into the flight computer

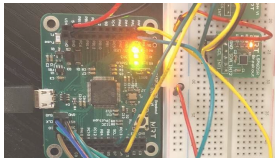
1: COMPONENT SELECTION

Choosing the appropriate component for the desired feature

Iterative Process for Every Component

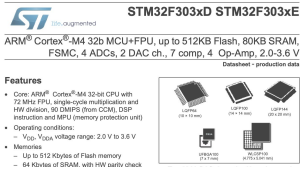
4: UNIT TESTING

Verifying that the hardware breakout and software drivers are functional simultaneously



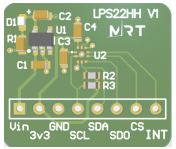
2: RESEARCH & LEARNING

Understanding the background needed to develop the component



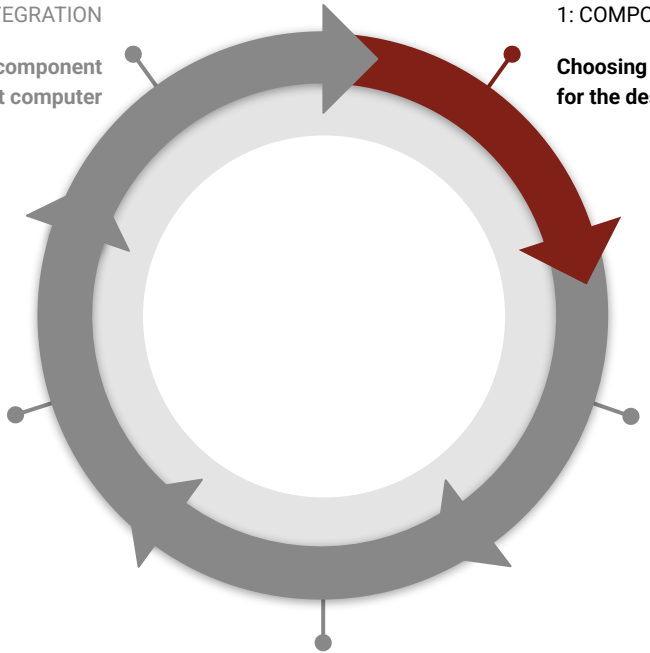
3: HARDWARE AND SOFTWARE DEVELOPMENT

Creating the hardware schematics and software drivers for the component




```
// LSM6DSR functions
extern stmdev_ctx_t lsm6dsr_init(void);
extern void get_acceleration(stmdev_ctx_t dev_ctx, float *acceleration_mg);
extern void get_angvelocity(stmdev_ctx_t dev_ctx, float *angular_rate_mdps);

// LPS22HH functions
extern stmdev_ctx_t lps22hh_init(void);
extern void get_pressure(stmdev_ctx_t dev_ctx,  float *pressure);
extern void get_temperature(stmdev_ctx_t dev_ctx,  float *temperature);
```

Step 1: Component & Feature Selection



1: COMPONENT SELECTION

Choosing the appropriate component for the desired feature

2: RESEARCH & LEARNING

Understanding the background needed to develop the component

3: HARDWARE AND SOFTWARE DEVELOPMENT

Creating the hardware schematics and software drivers for the component

4: UNIT TESTING

Verifying that the hardware breakout and software drivers are functional simultaneously

5: INTEGRATION

Integrating the component into the flight computer

Step 1: Component & Feature Selection

Telemetry Power Management Ejection

HARDWARE COMPONENT	PART NUMBER	SYSTEM
Inertial Measurement Unit	LSM9DS1 -> LSM6DSR	
Barometer	BMP280 -> LPS22HH	
Global Positioning System	NEO-M9N -> NEO-M8N	
Real-Time Clock	PCF8223 -> Built-in	
Radio	SX1262	
Logging	SD Card Reader	
STM32 Microcontroller	STM32F303RE	

FEATURES TO IMPLEMENT	SYSTEM
Integrated power management	
Solid-state ejection system	
Continuity check	
Current-sensing	
Task management/switching	
Variable logging frequency	
Special event messages	

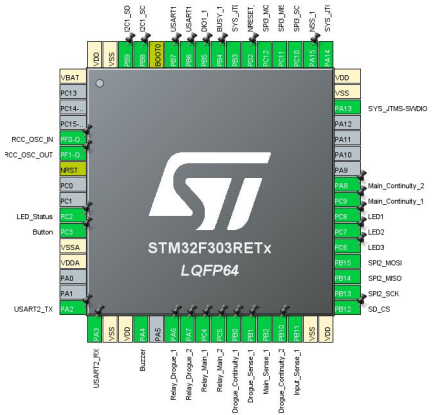
Microcontroller: STM32F303RE

Why STM32 series?

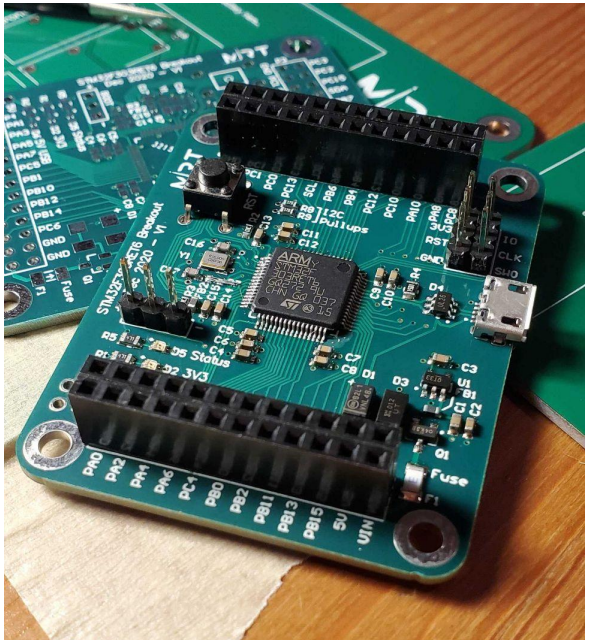
- Hardware and software **compatibility** between chips (easy to upgrade)
- **STM32Cube** development toolchain
- High customizability

Why STM32F303RE?

- Appropriate peripherals, middleware and communication interfaces
- Good CPU capability (72 MHz max) and power consumption



Breakout pin configuration

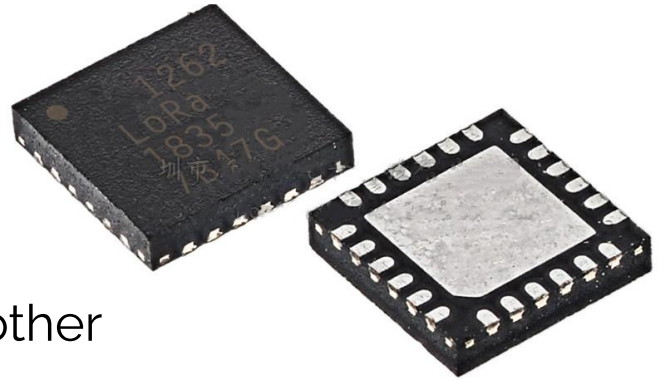


STM32F303RE breakout

Radio: SX1262 Transceiver

Objectives for transceiver

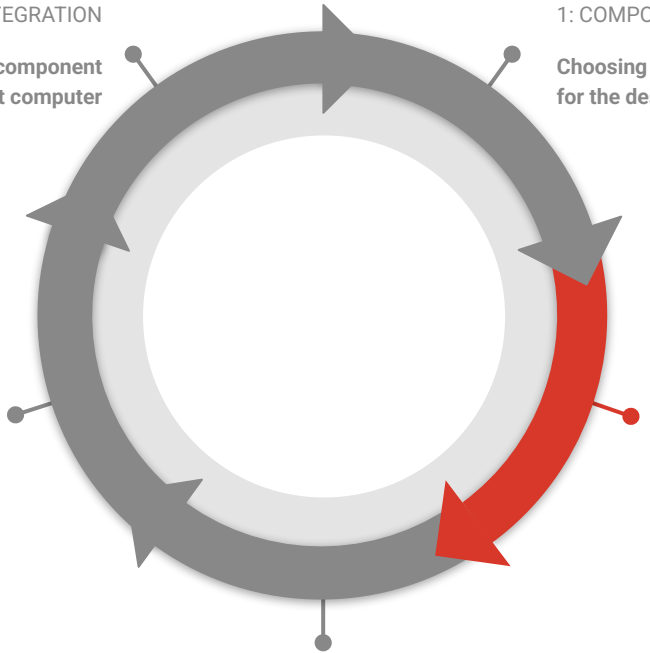
- Reduced **cost**
- Lower **power consumption**
- Allow for larger **link budget**



SX1262 was chosen after comparing several other solutions

- **~7%** of the cost of the previous year's radio (including parts for integration)
- **~1/8th** power consumption of previous year's radio
- Max link budget of **170 dB**
- **Robust** to noise and the Doppler effect

Step 2: Research & Learning



5: INTEGRATION

Integrating the component into the flight computer

1: COMPONENT SELECTION

Choosing the appropriate component for the desired feature

4: UNIT TESTING

Verifying that the hardware breakout and software drivers are functional simultaneously

2: RESEARCH & LEARNING

Understanding the background needed to develop the component

3: HARDWARE AND SOFTWARE DEVELOPMENT

Creating the hardware schematics and software drivers for the component

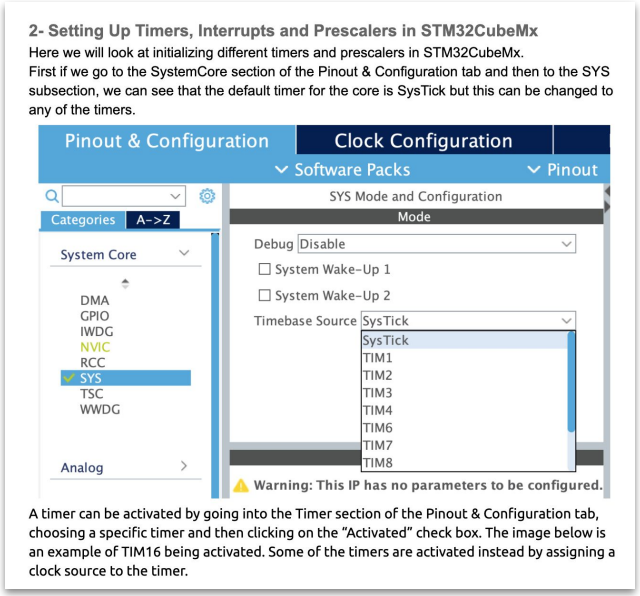
Learning STM32

Learning STM32Cube, APIs, HAL libraries

- *Peripherals* (e.g. ADC for current sensing, timers)
- *Memory* (e.g. FLASH reading and writing)
- *Middleware* (e.g. FatFS)
- *Communication protocols* (e.g. SPI for radio drivers)

Custom STM32/sensor schematics design

- Reading datasheets
- Learning about protocols (e.g. GPS NMEA format parsing)



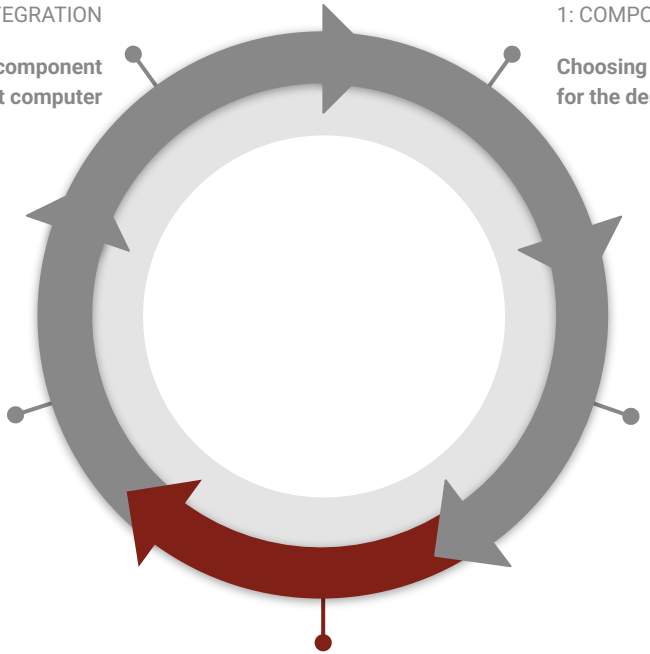
Example of documentation created for future reference

SRAD Radio Research

Optimizing radio transmissions

- **Reduce losses** from the PCB
  - Short traces
  - Impedance matching
  - Continuous ground plane for return currents
- Utilize a narrow band BPF for better **noise suppression**
  - Attenuate nearby transmitters to improve our reception
- Use of a TCXO and copper cutouts for **less frequency drift**
  - A still rocket in the desert sun can cause high temperatures
  - Other electronics create heat, especially the transmitter.

Step 3: Hardware & Software Development



1: COMPONENT SELECTION

Choosing the appropriate component for the desired feature

2: RESEARCH & LEARNING

Understanding the background needed to develop the component

3: HARDWARE AND SOFTWARE DEVELOPMENT

Creating the hardware schematics and software drivers for the component

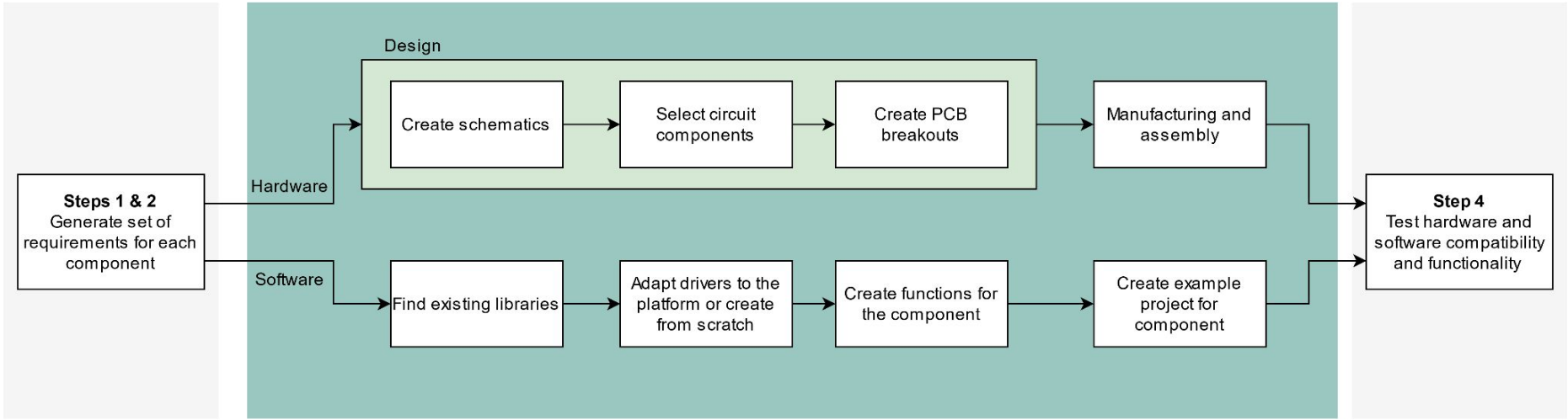
4: UNIT TESTING

Verifying that the hardware breakout and software drivers are functional simultaneously

5: INTEGRATION

Integrating the component into the flight computer

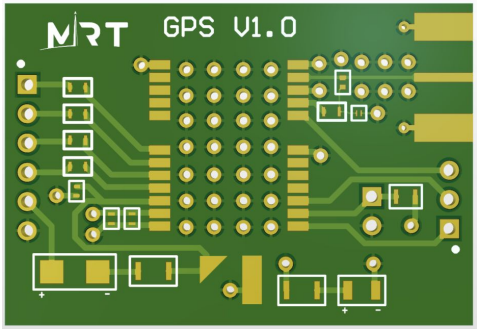
Step 3: Hardware & Software Development



Sensors and Components

Example: GPS (NEO-M8N)

- Hardware
  - Follow **datasheet** and existing commercial breakout **schematics**
  - Challenge: impedance matching
- Software
  - Create **custom drivers** based on Arduino GPS library
  - Challenges:
    - Parsing data from NMEA format
    - Checksum



GPS breakout

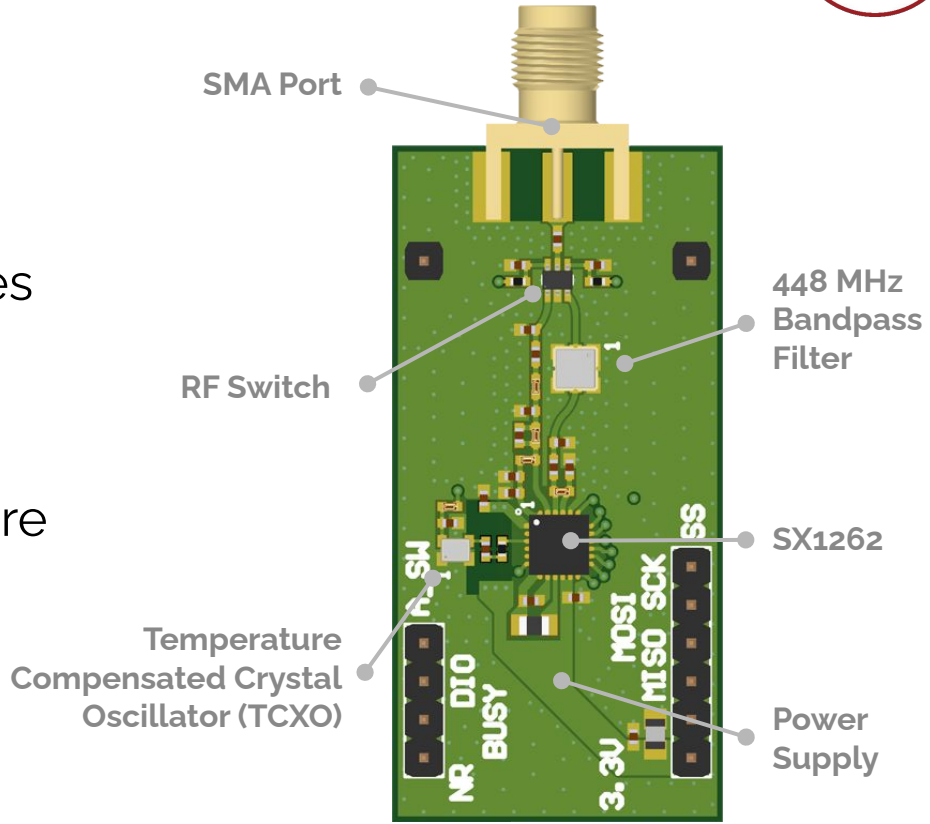
```c
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_USART1_UART_Init();
MX_USART3_UART_Init();
/* USER CODE BEGIN 2 */

float latitude;
float longitude;
float time;

char buffer[100];

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
GPS_Poll(&latitude, &longitude, &time);
sprintf(buffer, "(Latitude, Longitude): %f,%f   (Time): %f\n", latitude, longitude, time);
HAL_UART_Transmit(&huart3, (uint8_t*)&buffer, 100, 100);
memset(buffer, 0, sizeof(buffer));
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
```

GPS example project

SRAD Radio - Hardware

Design process

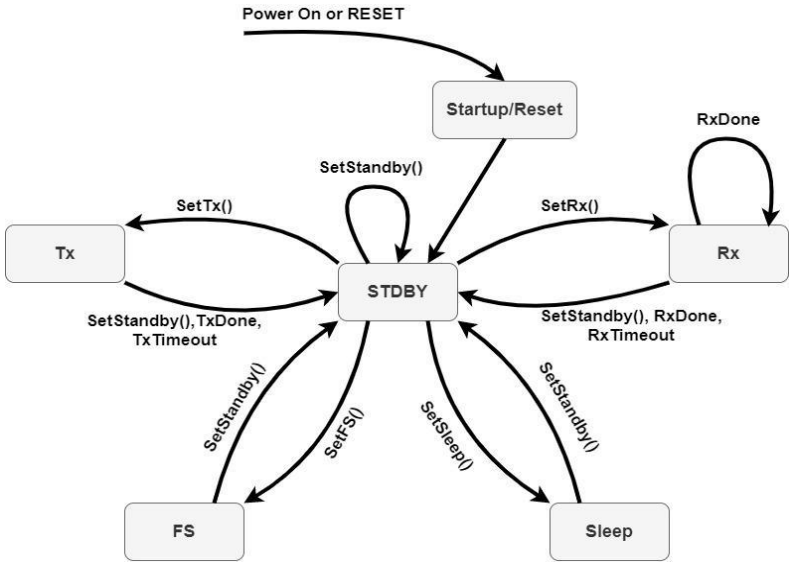
1. Initial design from **reference documents** and desired changes
2. **Feedback** from professors, TAs and the manufacturer
3. If not satisfied, **redesign** hardware layout and repeat step 2



SRAD Radio - Software

Development of software library

- Written and prototyped in **Arduino**
- 3 main operational modes
  - Standby: Change operating parameters
  - Tx: Transmit packets
  - Rx: Receive packets
- **Instructions** = operational code and parameters
- Converted to **STM32** library

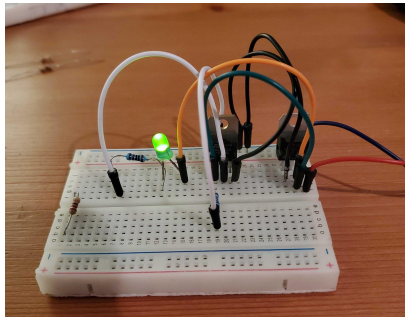


SX1262 state diagram

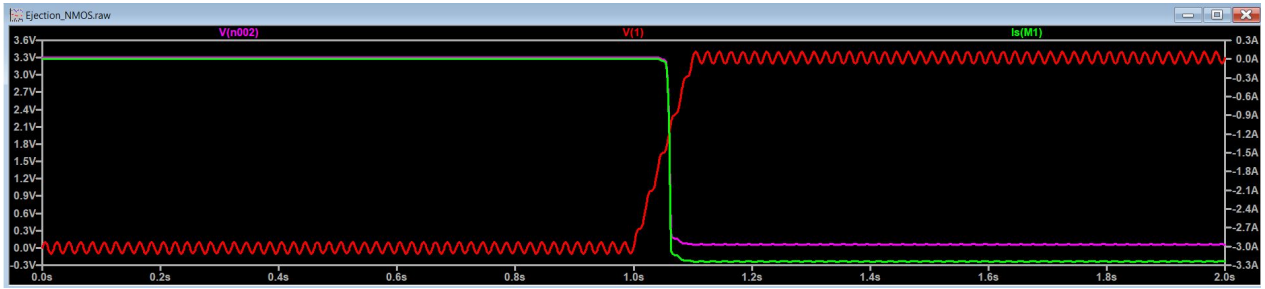
Ejection Circuitry

Design Process

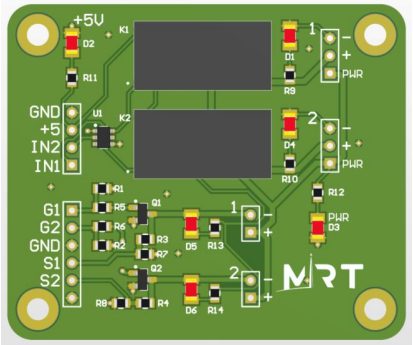
- Prototyped and simulated in **LTspice**
- **Breadboard** for proof-of-concept
- **Breakout** for testing with other breakouts
- **Integration** into Flight Computer



Breadboard prototype



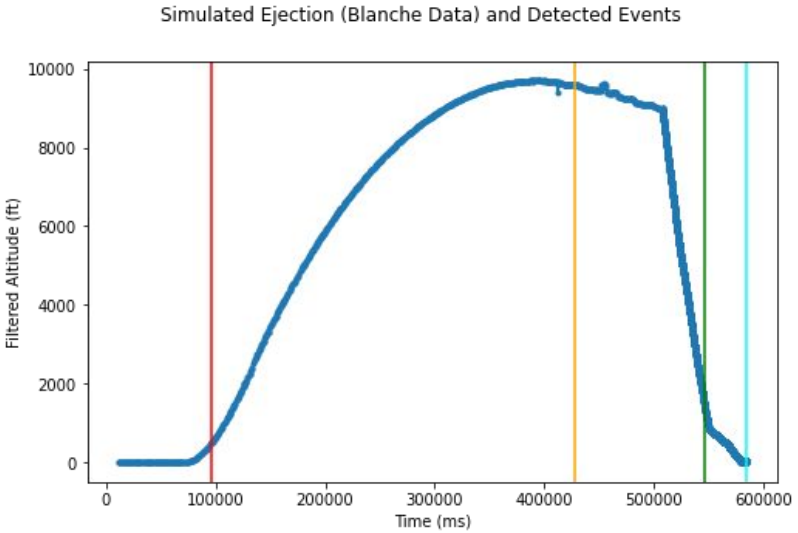
LTspice simulation results



PCB breakout

Ejection Algorithm Design

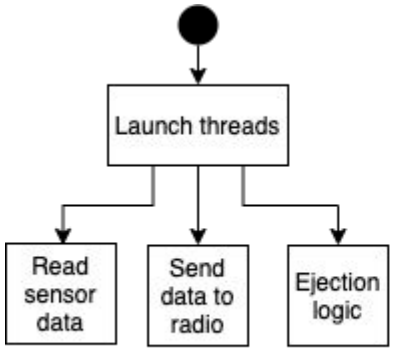
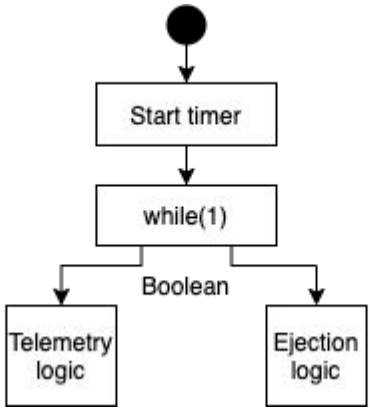
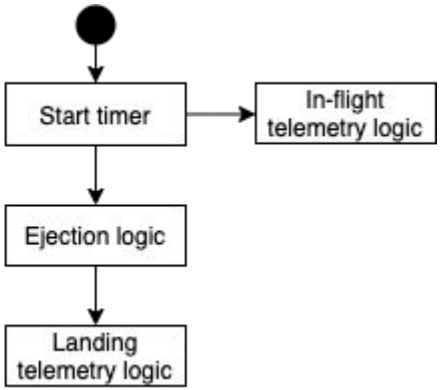
- Starting point: previous implementations
- **Simulate** performance in **Python** using previous flight data
- **Refine** apogee detection:
  - Linear regression on altitude history to detect zero velocity crossing point
- **Translate** back to **STM32** and create example project



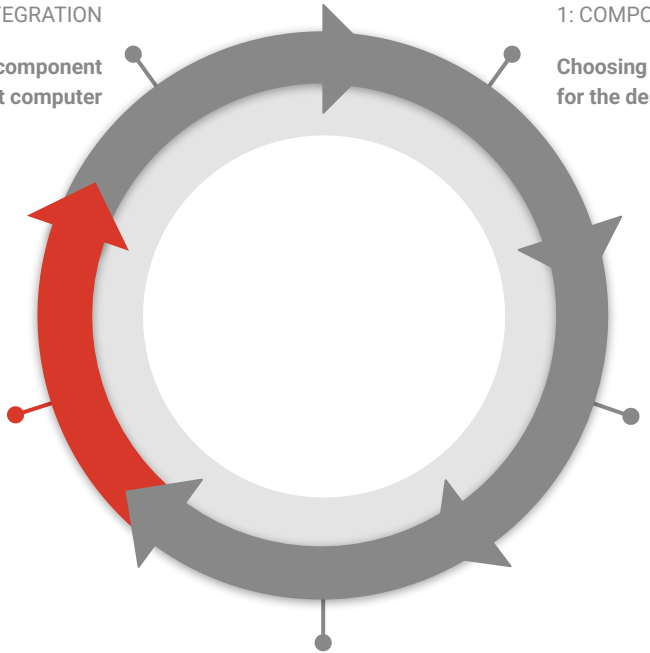
Detection Events:

- Launch
- Apogee
- Main Deployment Altitude
- Landing

Task Management

Real-Time Operating System (FC V1)	State Machine (Not developed)	Interrupt-Driven (FC V2)
• Specialized **threads** controlling each function • Execution based on thread **priorities**	• Timer **interrupt** event controls execution (switch between telemetry and ejection)	• *Before landing*: main **ejection** logic, telemetry in timer callback • *After landing*: **telemetry** in main loop
		

Step 4: Unit Testing



5: INTEGRATION

Integrating the component into the flight computer

1: COMPONENT SELECTION

Choosing the appropriate component for the desired feature

4: UNIT TESTING

Verifying that the hardware breakout and software drivers are functional simultaneously

2: RESEARCH & LEARNING

Understanding the background needed to develop the component

3: HARDWARE AND SOFTWARE DEVELOPMENT

Creating the hardware schematics and software drivers for the component

Step 4: Unit Testing

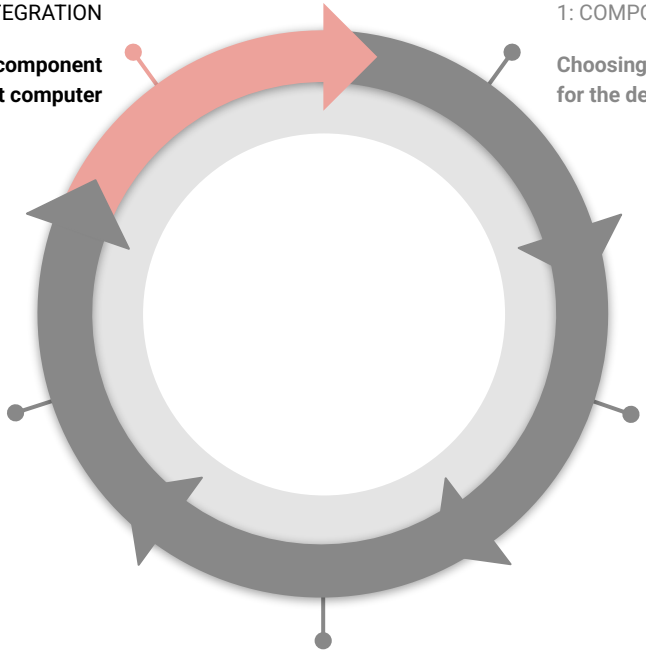
Purpose: Ensure that each component functions **by itself**

Tests

- Run example project created with one component at a time
- Function **runtime** in software (using ITM trace)

Function	Peripheral	Clock Speed (MHz)	Avg. Clock Cycles	Avg. Times (s)
ADC measurements	ADC3, ADC4	64	3625.7	0.0000566515525
Continuity check	GPIO	64	224.91	0.00000351422
RTC date + time	RTC	0.04	538.89	0.00000842016
Pressure + temperature	I2C1	64	53802.91	0.00084067047
Acceleration + Angular velocity	I2C1	64	54719.11	0.00085498609
GPS Poll	SPI1	64	7924066.38	0.1238135344
String formatting	Processor	64	68528.36	0.00107076
Radio transmission (both parts)	SPI2	64	101884558.7	0.19194625
Print to computer	USART1	32	1534756.76	0.02398056

Step 5: Integration



5: INTEGRATION

Integrating the component into the flight computer

1: COMPONENT SELECTION

Choosing the appropriate component for the desired feature

2: RESEARCH & LEARNING

Understanding the background needed to develop the component

3: HARDWARE AND SOFTWARE DEVELOPMENT

Creating the hardware schematics and software drivers for the component

4: UNIT TESTING

Verifying that the hardware breakout and software drivers are functional simultaneously

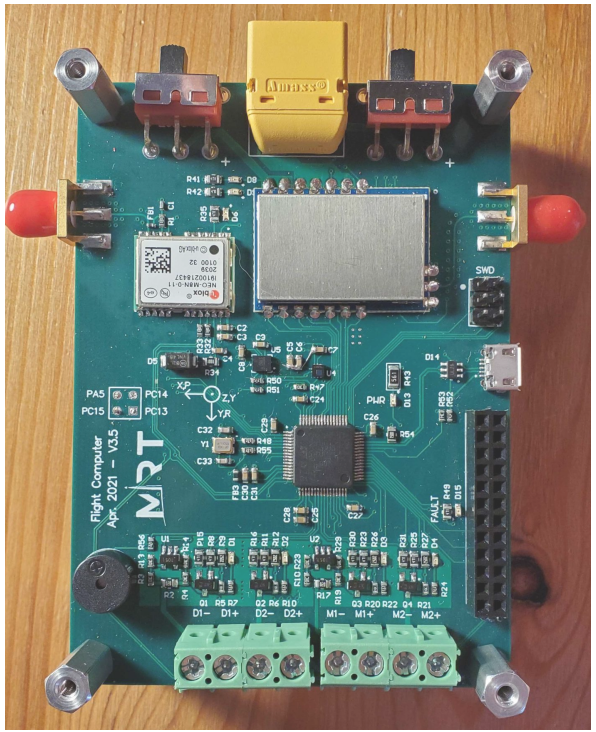
Step 5: Integration - Hardware

Specifications

- Breakouts were merged together into a single PCB
- **4-layer** stackup (RF/signal, ground, power (3.3 V), signal)
- Board dimensions: **70.87 x 97.79 mm / 2.79 x 3.85 in**
- Battery input voltage range: **6 - 15 V**
- Average current consumption: **120 mA**
- Weight: **67.8 g / 2.39 oz**

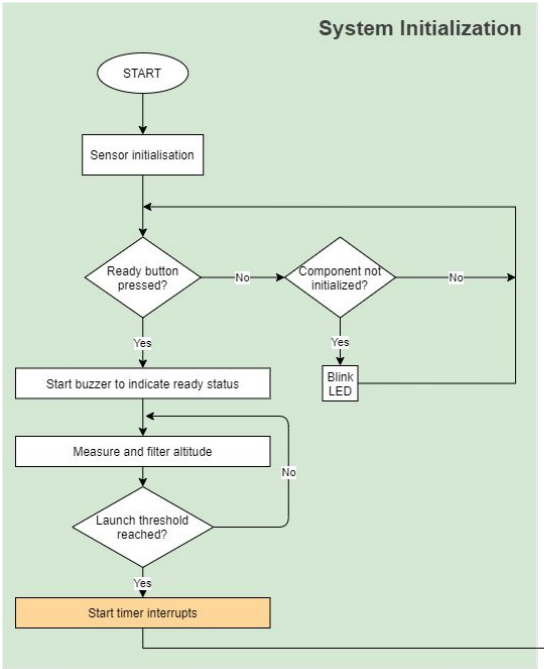
Notable features

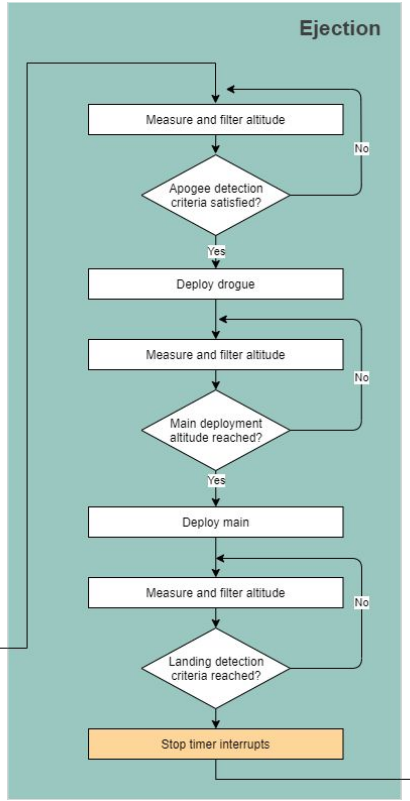
- USB
- Debug headers
- SMA connectors for radio and GPS
- XT60 power connector



Assembled Flight Computer

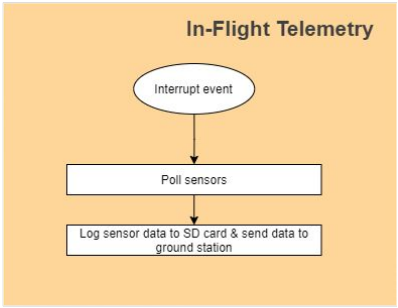
Step 5: Integration - Software

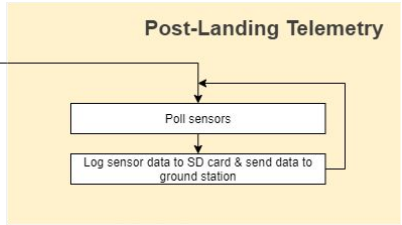




Telemetry string format:

`S,ACCx,ACCy,ACCz,MAGx,MAGy,MAGz,PRESSURE,LAT,LONG,HOUR,MIN,SEC,SUBSEC,E\n`





- Data **transmitted** to Ground Station every 2 s
- Data **saved** to SD card every 0.5 s during flight and 2 s when idle

System Testing

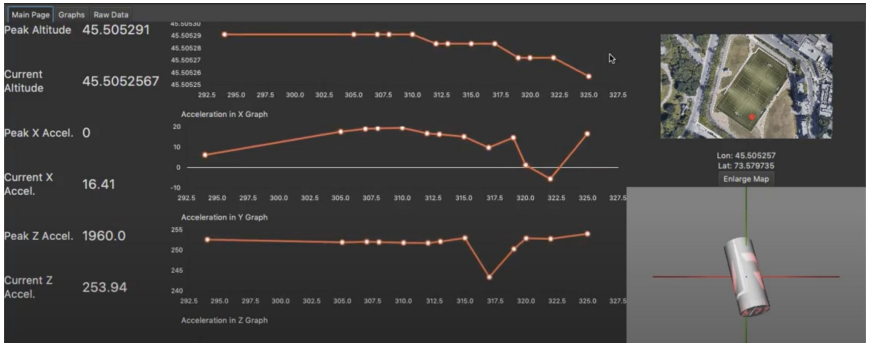
Telemetry Functionality Testing (1)

Objectives

- Verify that altitude, accelerometer/gyroscope, and GPS **sensors** read data
- Characterize SRAD **radio** performance

Tests Performed

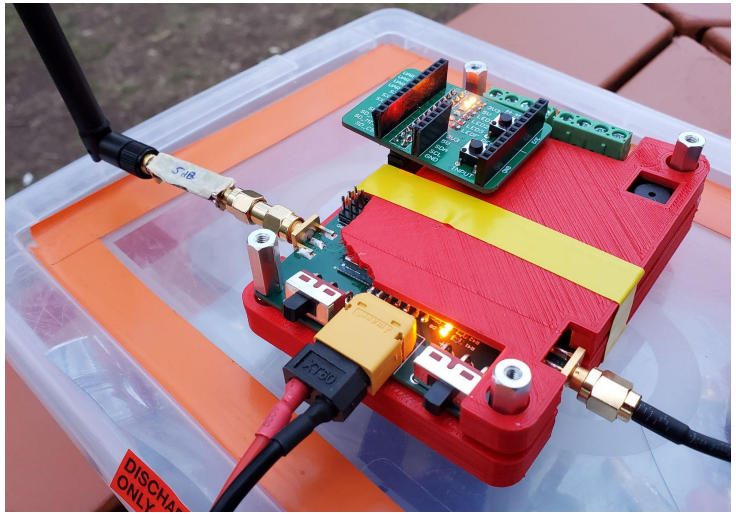
- Short range outdoors (200 m / 656 ft)
- Simulated long range with attenuators
- Actual long range outdoors (10 km / 6.21 miles)



Ground station GUI plot of FC data

Assembled Flight Computer

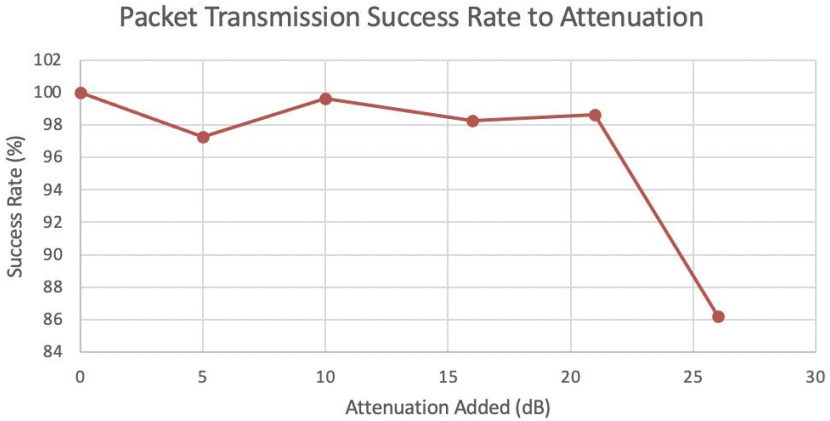
Telemetry Functionality Testing (2)



Simulated long range test setup with attenuators

Test Setup Parameters

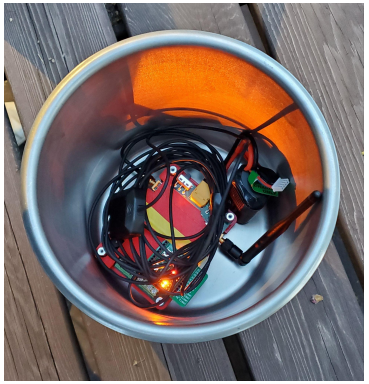
Transmit Gain (dBi)	0.7	Frequency (MHz)	433
Receive Gain (dBi)	11.5	Initial distance (ft)	328



Ejection Functionality Testing

Objectives

- Verify **ejection algorithm** detects events
- Reproduce simulation results
- Evaluate **robustness** of algorithm to different flight profiles

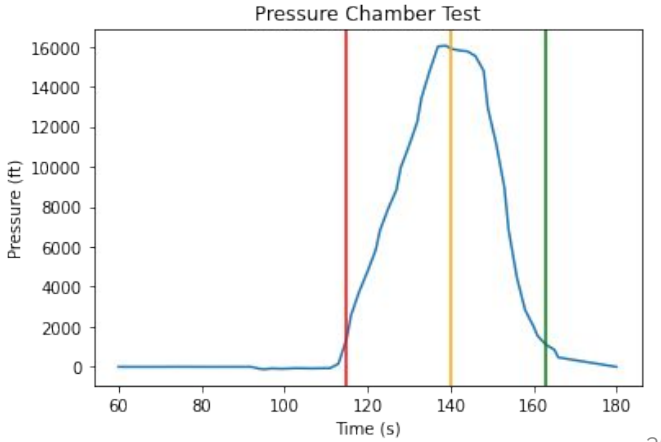


Tests

- Simulated flight test
- Vacuum chamber tests
  - Verified at different apogees

Detection Events:

- Launch
- Apogee
- Main Deployment Altitude



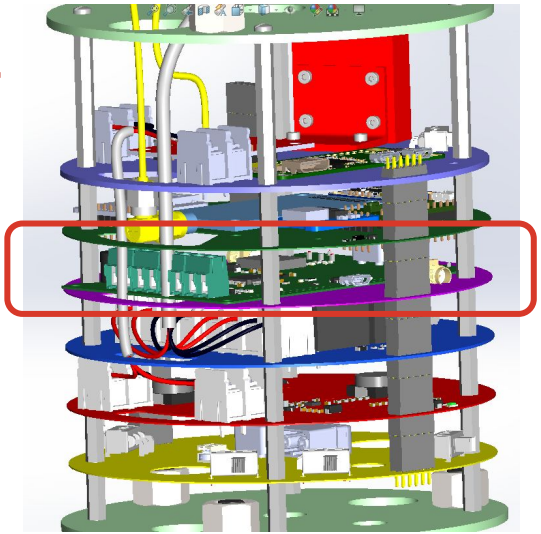
Testing at SAC

Objectives

- Verify sensors read data inside a **fully-assembled rocket**
- Ejection software **detects events**
  - Launch detection, apogee detection, etc.

Expectations

- Telemetry data stored to SD card
- All events successfully detected and logged



Flight Computer in AV Bay CAD

Follow-On Work

Future Improvements - General (1)

Summer 2021/Next design cycle improvements

- Dynamic file names for storage
- freeRTOS **multi-threading**
- **FLASH storage** to replace / complement SD storage
- Improved power management
- More sensors (e.g. magnetometer)

Other possible improvements

- Upgraded STM32 chip
- Absorption of video recorder

Future Improvements - Specific (2)

SX1262 improvements

- Add **power amplifier** to bring transmitter output to 1 W
- Add more vibration/acceleration tolerant oscillator to **mitigate frequency drift**
- Thorough **testing** with lab equipment (spectrum analyzer and EMI testing)

Telemetry improvements

- Implementing **half-byte encoder**
- **Kalman filtering** for sensor data

Q&A

Thank you for listening!