This report shall document all the preparatory procedures conducted for the SRAD Motor test in October 2024, including:

1. Components used and their respective wiring
2. Code implemented along with its explanation
3. Errors encountered along with their mitigation measures
4. Possible points of failure
5. Precautionary measures

The following document shall be divided into 2 main sections for ease of reference, viz.

1. Electronics
2. Computation

**Electronics:**

The electronics employed for the SRAD motor test constitute the hardware required for the following purposes:  
1) Wireless ignition of the motor

2) Data acquisition and logging

The components used are:

1. Button Load Cell of 1tf capacity (1)
2. Weight indicator (1)
3. Arduino UNO R3 (2)
4. RYLR998 LoRa module (2)
5. D4184 MOSFET module (2)
6. SD Card- 16GB (2)
7. SD Card reader (2)
8. Lithium-Polymer (LiPo) battery- 7.4V 6200mAh 60C (2)
9. Ignitors (2)
10. Resistors (2)
11. Toggle switches with cover (2)
12. Breadboard (1)
13. PDB
14. Jumper Wires

The entire setup has been divided into 2 for the purpose of operations and reference:

1. Testbed/motor side
2. Ground side

**Testbed/Motor side:**

The motor is situated on the testbed, where it is placed on a button load cell. The motor is fired using an ignitor. The components on the Testbed side are as follows:

1. Button Load cell (UNV-C 1tf) (1)
2. Weight indicator (TP400 mini) (1)
3. RYLR998 LoRa module (1)
4. D4184 MOSFET module (2)
5. SD Card- 16GB (1)
6. SD Card adapter (1)
7. Lithium-Polymer (LiPo) battery- 7.4V 6200mAh 60C (2)
8. Ignitors (2)
9. Arduino UNO R3 (1)
10. PDB (1)

The load cell model is UNV-C 1tf, which means it can withstand a maximum load of 1 ton. The load cell data is transmitted to a weight indicator, which has a digital display. The weight indicator model is TP400mini. The load cell is connected to the weight indicator via a cable with 5 wires. The 5 wires are:

1. SHIELD, a bare wire
2. SENSE+, the green colour wire
3. SENSE-, the white colour wire
4. EXC+, the red colour wire
5. EXC-, the black colour wire

The cable has a 5-pin connector that goes to the load cell side and fits only in one particular configuration. The 5 wires on the other end are connected to the screw terminals of the weight indicator in the following fashion:

1. SHIELD to terminal 3
2. SENSE+(green) to terminal 4
3. SENSE-(white) to terminal 5
4. EXC+(red) to terminal 6
5. EXC-(black to terminal 7)

The weight indicator is powered from the AC Mains supply via a 24V AC-DC adapter, as recommended by the data sheet. The connections are as follows:

1. Red wire (24V) to terminal 1
2. White wire (Ground) to terminal 2

For the purpose of transferring the load cell data to an Arduino UNO R3 at a very quick rate for data analysis, the analog output pins of the weight indicator are used. The analog output pins are connected to the Arduino in the following manner:

1. Pin 13 ANALOG(+) is connected to pin A0 of Arduino
2. Pin 14 ANALOG(-) is connected to GND of Arduino (via PDB)
3. TxD is connected to terminal 9

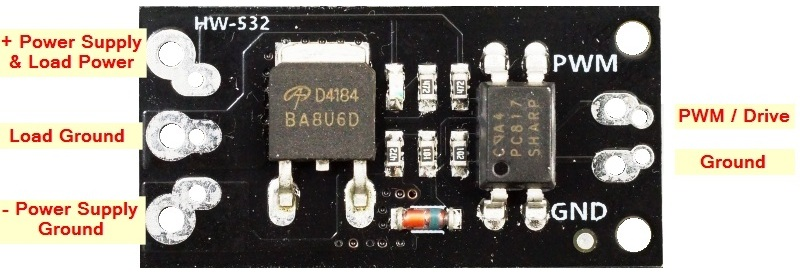
A Power Distribution Board (PDB) is used to distribute the power from the Arduino from a central location. The PDB consists of 6 2-pin screw terminals, with each screw terminal providing a place to connect 5V and Ground (GND). All components on the testbed side require 5V, and GND are henceforth connected to the PDB.

An RYLR998 LoRa module is used to transmit and receive data and commands to and from the Ground side. The required connections are as follows:

1. VDD to 3.3V of Arduino
2. RST is left unconnected
3. RxD to pin 6 of Arduino
4. TxD to pin 7 of Arduino
5. GND to GND of Arduino

The motor is ignited using an E-match (an ignitor). 2 ignitors are being used for redundancy purposes. The ignitors are triggered using 2 D4184 MOSFET modules, one for each ignitor. They will be named D4184-A and D4184-B, respectively, for ease of reference. The connections to the D4184s are as follows:

1. PWM of D4184-A to pin 4 of Arduino
2. PWM of D4184-B to pin 5 of Arduino
3. GNDs of both D4184s are connected to GND of Arduino (via PDB)
4. One side of each of the ignitors is connected to Load power of each of the D4184s
5. The other side of the ignitors is connected to Load ground of each of the D4184s
6. + Power supply of each of the D4184s is connected to the positive terminal of the LiPo battery using T-connectors
7. Power supply ground of each of the D4184s is connected to the negative terminal of the LiPo battery using T-connectors



All data coming from the load cell is logged into a 16GB SD card via an SD Card adapter. The connections to the SD Card adapter are as follows:

1. CS to pin 10 of Arduino
2. SCK to pin 13 of Arduino
3. MOSI to pin 11 of Arduino
4. MISO to pin 12 of Arduino
5. VCC to 5V of Arduino (via PDB)
6. GND to GND of Arduino (via PDB)

**Ground side:**

The Ground side contains the physical implementation of the Finite State Machine (FSM). The components used on the Ground side are as follows:

1. Arduino MKR Zero (has internal SD card holder) (1)
2. RYLR998 LoRa module (1)
3. Toggle switches with cover (2)
4. Breadboard (1)
5. SD Card adaptor (1)

The RYLR998 LoRa module is connected to the Arduino MKR Zero to send and receive instructions/data. The connections are as follows:

1. VDD to 3.3V of Arduino MKR Zero
2. RST is left unconnected
3. RxD to pin 14 of Arduino MKR Zero
4. TxD to pin 13 of Arduino MKR Zero
5. GND to GND of Arduino MKR Zero

There are two switches used on the Ground side. One is for ARM and one is for Launch. These switches are Single Pole Double Throw (SPDT) toggle switches with cover. They each have 3 terminals. The connections are as follows:

1. ARM Switch:
2. VCC (Red wire) to 5V of Arduino MKR Zero
3. GND (Black wire) to GND of Arduino MKR Zero
4. Common (Green wire) to Pin 2 of Arduino MKR Zero

*\*\*22k pull down resistor between GND and Common*

1. LAUNCH Switch:
2. VCC (Red wire) to 5V of Arduino MKR Zero
3. GND (Black wire) to GND of Arduino MKR Zero
4. Common (Green wire) to Pin 3 of Arduino MKR Zero

*\*\*22k pull down resistor between GND and Common*

**Errors encountered and the solutions implemented:**

The spreadsheet attached below contains the log of daily errors that was encountered during the various tests that were conducted:

[Motor test prep log](https://docs.google.com/spreadsheets/d/1KKgFlg0Wpx8H-zzZQhJUoz4TRmRtyMhWwGH2FQauR-8/edit?gid=0#gid=0)

**Computation:**

The computation aspect of the Motor test consists of the code employed to control the electronics appropriately to achieve wireless ignition, communication between the ground side and the testbed/motor side, data acquisition and logging.

**Testbed/Motor side Code:**

The testbed side was getting data from the load cell through an RS232 TTL converter, but this caused a very low ODR (Output Data Rate), hence we eliminated the RS232 and are taking analog data from the weight indicator.

We have a function to find the calibration factor to calibrate the weight with a known weight before setup of the microcontroller, and map the voltage to kilograms.

The succeeding functions parse incoming data from load cell, log data into a text file (“loadcell.text”) and transmit data to the ground side using the RYLR-998 on both sides (through AT commands).

Problems we faced:

Initially, the log file was named ‘RangeTest.txt’ and though comm was established between ground and motor side RYLRs, it said ‘Unable to open log file’.

It did not work with the file name ‘Range.txt’ either.

It only worked with ‘loadcell.txt’.

Possible reasons:

8.3 filename format. According to this format, the filename should be upto 8 characters and and the extension should be 3 characters. Though ‘Range.txt’ fits this format, ‘loadcell.txt’ is exactly 8 characters and three characters respectively.

parseRYLR() - extracts data from the string input from RYLR.

The format in which the data is received is: +RCV=<address>,<length>,<data>,<rssi>,<snr>

Where:  
 <address> transmitter id ;  
<length> data length  
<data>ASCII format data  
<RSSI> received signal strength indicator  
<SNR> Signal to noise ratio  
(example: 50,5,HELLO,-99,40)

sendState() - broadcasts the state (ADDRESS=0) to all RYLRs in range, the current state of the FSM (SAFE, ARMED, or LAUNCHED)

Ground Station:

At the ground station we have a similar code that transmits and receives data to and from the motor side through RYLR 998 modules. When ARM/LAUNCH signals are sent to the motor side, the current state is received back at the groundside and is printed.

FSM:

The FSM (Finite State Machine) that we’ve implemented in the code is to ensure that the motor is ignited only when the switches are turned on in the order ARM and LAUNCH, i.e the state change is SAFE->ARM->LAUNCH. This triggers launch of rocket (or ignition of charge in this case) and ensures safety. In line with the rules of the SA cup, the launch is triggered by one event only.