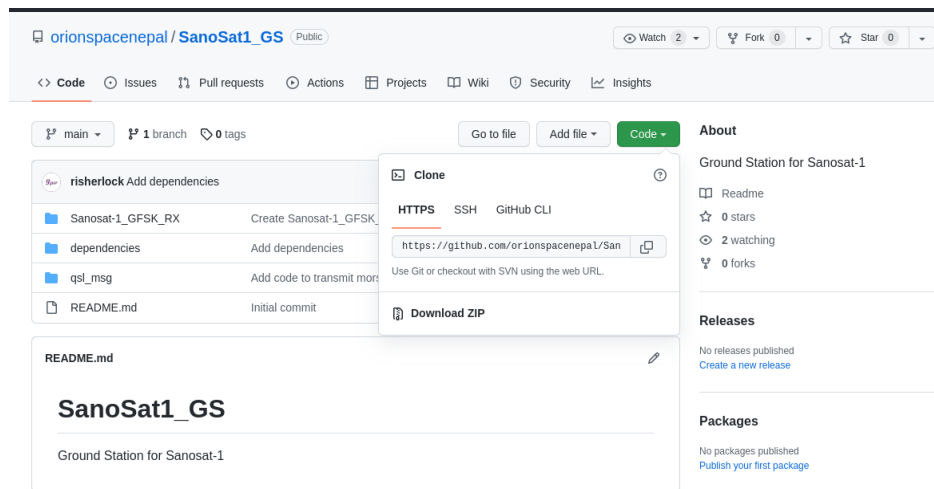


Setting up Arduino IDE Environment

1. Make sure the Arduino IDE is installed in your computer. You can install Arduino IDE from here:
<https://www.arduino.cc/en/software>
2. Goto https://github.com/orionspacenepal/SanoSat1_GS and download the repository on your computer by clicking **Code** and then **Download ZIP**.

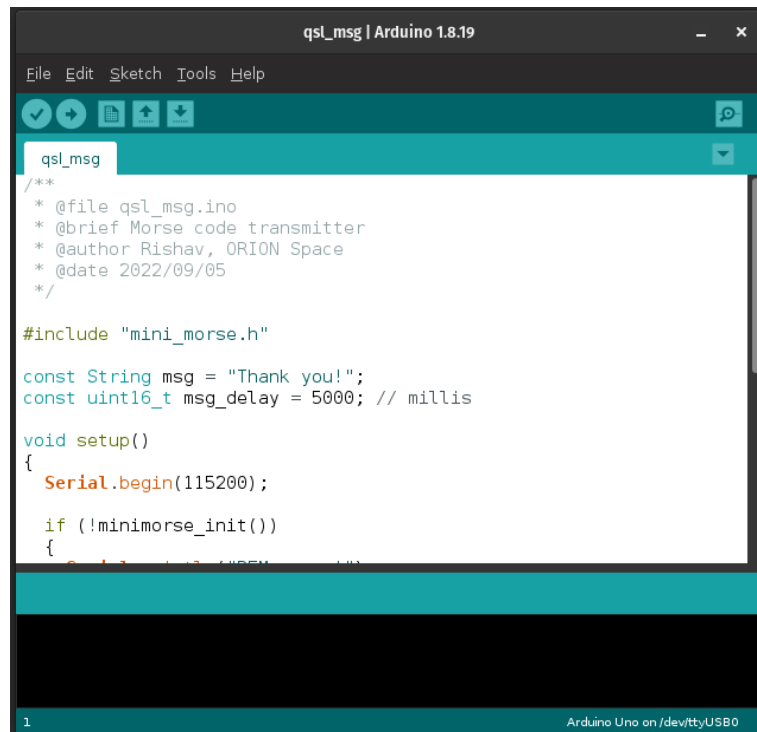


3. UnZIP the directory.
4. In the repository you will see two directories:
 - a. **qsl_msg** - contains a code to transmit morse code of your choice using the RFM26 radio module.
 - b. **Sanosat-1_GFSK_RX** - contains a code to use the board as a ground station to the SanoSat-1 satellite.

NOTE: Now that we have Arduino IDE and codes, we are ready to upload code to the board. This document will take an example of **qsl_msg**, but the same applies for other codes too.

Flashing Firmware to the Ground Station

1. Open the `.ino` file of the software you want to upload. For our example, we will consider `qsl_msg.ino` inside the `qsl_msg` directory. The interface looks like the one shown in figure below.



```
qsl_msg | Arduino 1.8.19
File Edit Sketch Tools Help
qsl_msg
/**
 * @file qsl_msg.ino
 * @brief Morse code transmitter
 * @author Rishav, ORION Space
 * @date 2022/09/05
 */

#include "mini_morse.h"

const String msg = "Thank you!";
const uint16_t msg_delay = 5000; // millis

void setup()
{
  Serial.begin(115200);

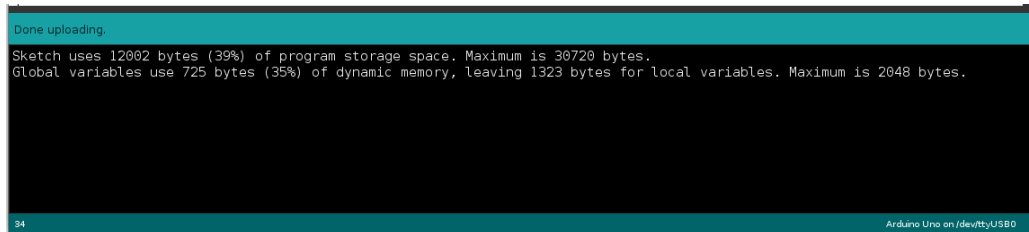
  if (!minimorse_init())
  {
    // ...
  }
}

1 Arduino Uno on /dev/ttyUSB0
```

2. Follow the following steps to configure the microcontroller and port to upload code on the Ground Station board.
 - a. To configure microcontroller, goto:
`Tools` -> `Board` -> `Arduino Pro or Pro Mini`, and
`Tools` -> `Processor` -> `ATmega328P (3V3, 8MHz)`
 - b. To configure port, connect the Ground Station to a USB port
goto:
`Tools` -> `Port` -> `Serial ports` -> `COM5`

NOTE: COM5 above is just for example, it could be COM17 or any other number. If there are multiple ports on option, you can find the one corresponding to the Ground Station by plugging it off and noting the COM port that disappears from the list. Now you can reconnect the board and follow step 2b.

3. Now, all is good, goto **Sketch** -> **Upload** to flash the code to the Ground Station. Following message shows that the uploading is successful.



```
Done uploading.  
Sketch uses 12002 bytes (39%) of program storage space. Maximum is 30720 bytes.  
Global variables use 725 bytes (35%) of dynamic memory, leaving 1323 bytes for local variables. Maximum is 2048 bytes.
```

4. Hook up the CW receiver and enjoy the morse message. If you don't have a radio receiver laying around, you can enjoy the same morse code in the Serial Monitor. Output will be something like the image below.



```
/dev/ttyUSB0  
Morse radio is ready!  
..... - - - - -
```

You can copy this message and paste it to <https://morsedecoder.com/> to view the message.

Software Configurations and Debug

This part of the document contains details of some essential configurations and ways to know the status of the device based on the LED (D9 in the board).

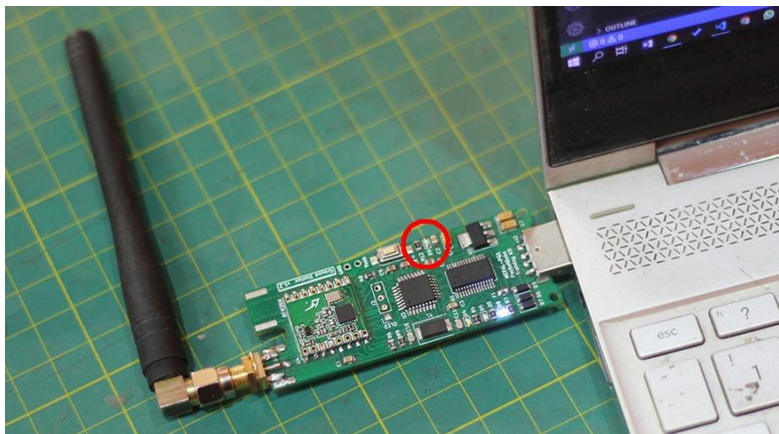
1. Transmission frequency and the WPM (Words Per Minutes) for Morse code can be configured in `mini_morse.cpp`.
 - a. Transmission frequency -> `MINI_MORSE_RFM_TX_FREQUENCY`
 - b. Words Per Minute -> `MINI_MORSE_WPM`

```

10
11  /* Start configuration */
12
13  // Morse transmission configurations
14  #define MINI_MORSE_RFM_TX_FREQUENCY 436.235
15  #define MINI_MORSE_WPM 20
16

```

2. LED (D9 in the board) state for debug. It is located in a red circle.



- i. LED flashes and stays on forever --> Radio module error
- ii. If the radio module is working fine, then the LED stays on during the time of CW transmission only. It is off between two message transmissions (`msg_delay` in the code `qsl_msg.ino`). This will allow the user to know that it's transmitting morse codes when the LED is on. During transmission, you can also see the D4 in the board flashing.
