FOSSA Systems

Richard Bamford

FOSSA Systems, Advisory-LTD

<http://richardbamford.dev>

rbamfordz@gmail.com

FOSSASAT-1 Software Report

2 - Software Design Report

# Changelog

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author(s) | Confirmation Signature(s) | Description |
| 03/05/2019 | Richard Bamford |  | Created initial list of systems to detail. |
| 05/05/2019 | Richard Bamford |  | * Communication system description/Brief. * Added appendix and bibliography. * Added main Arduino setup and loop flow charts. |
| 05/05/2019 | Richard Bamford |  | Changelog added |
| 08/05/2019 | Richard Bamford |  | * Added more detail to the protocol definition section. * Added safety & security headings. * Fixed figure numbers. * Added cover page. * Added headings. |
| 14/05/2019 | Richard Bamford |  | * Deployment system chapter and diagrams. |
| 16/05/2019 | Richard Bamford |  | * Completed the document ready for proof reading |
| 17/05/2019 | Richard Bamford, Gary Bamford |  | * Added Brief instead of introduction titles. * Added figures to contents page. * Added introduction. * Removed unused STATE flag and messages. * Added numbers to headings. * Added introduction to sections. * Simplified the communications system section. |
| 22/05/2019 | Richard Bamford |  | * Edits to wording. |
| 29/05/2019 | Richard Bamford,  Julian Fernandez |  | * Defined 2 new commands that signal to the satellite to transmit a message using SF7 option. |
| 30/05/2019 | Richard Bamford, Julian Fernandez |  | * Defined the new commands to switch between SF11 and SF10. |
| 03/06/2019 | Richard Bamford, Julian Fernandez |  | * Low power mode. * MPPT disable/enabled command. * Low power mode disable/enable command * Changed system information packet to start with low power mode boolean |
| 05/06/2019 | Richard Bamford |  | * MPPT enable and disable command changed to temperature bypass. |
| 10/06/2019 | Richard Bamford |  | * Added saving and loading configuration of power control system. * Added command to enable and disable MPPT circuits completely. |

Contents

[1 Changelog 1](#_Toc10465369)

[2 Introduction 6](#_Toc10465370)

[2.1 Authors note 6](#_Toc10465371)

[3 Documentation references 7](#_Toc10465372)

[3.1 Main Arduino Program 7](#_Toc10465373)

[3.2 Communications System 7](#_Toc10465374)

[3.3 Safety & Security System 7](#_Toc10465375)

[3.4 Deployment System 7](#_Toc10465376)

[3.5 Battery Conservation System/Automatic Interval Control System 7](#_Toc10465377)

[3.6 Battery Temperature Control System 7](#_Toc10465378)

[3.7 Configuration System 7](#_Toc10465379)

[3.8 Hardware Interface System 7](#_Toc10465380)

[3.9 System Information System 7](#_Toc10465381)

[4 Main Arduino Program 8](#_Toc10465382)

[4.1 Background 8](#_Toc10465383)

[4.2 Setup Function 8](#_Toc10465384)

[4.2.1 Description 8](#_Toc10465385)

[4.3 Looping function 9](#_Toc10465386)

[4.3.1 Description 9](#_Toc10465387)

[5 Communication System 10](#_Toc10465388)

[5.1 Background 10](#_Toc10465389)

[5.2 Architecture 10](#_Toc10465390)

[5.3 Program flow 11](#_Toc10465391)

[5.4 Flag Definitions 12](#_Toc10465392)

[5.5 Protocol Definition 12](#_Toc10465393)

[5.5.1 Protocol Description 12](#_Toc10465394)

[5.5.2 Protocol Lookup Table 12](#_Toc10465395)

[6 Safety & Security System 14](#_Toc10465396)

[6.1 Background 14](#_Toc10465397)

[6.2 RadioLib (SX1278) error codes 14](#_Toc10465398)

[6.3 EEPROM deployment fault checking. 15](#_Toc10465399)

[6.3.1 introduction 15](#_Toc10465400)

[6.3.2 EEPROM fault modes 15](#_Toc10465401)

[7 Deployment System 16](#_Toc10465402)

[7.1 Background 16](#_Toc10465403)

[7.2 Sequence Control 16](#_Toc10465404)

[7.3 Debugging mode 17](#_Toc10465405)

[7.4 Integration mode 17](#_Toc10465406)

[7.5 Launch mode 17](#_Toc10465407)

[7.6 Deployment Sequence 18](#_Toc10465408)

[9 Battery Conservation System 19](#_Toc10465409)

[9.1 Background 19](#_Toc10465410)

[9.2 Low Power mode 19](#_Toc10465411)

[10 Battery Temperature Control System 19](#_Toc10465412)

[10.1 Background 19](#_Toc10465413)

[11 Appendix A – Diagrams 20](#_Toc10465414)

[11.1 Arduino sketch setup function flowchart 20](#_Toc10465415)

[11.2 Arduino loop function flow chart 21](#_Toc10465416)

[11.3 Communication system program flow chart 22](#_Toc10465417)

[11.4 Deployment sequence control flow chart 23](#_Toc10465418)

[11.5 Deployment sequence flow chart 24](#_Toc10465419)

[12 References 25](#_Toc10465420)

[Figure 1 - Arduino sketch setup flow chart 18](#_Toc10465421)

[Figure 2 - Arduino sketch loop function 19](#_Toc10465422)

[Figure 3 - Communications system flow chart 20](#_Toc10465423)

[Figure 4 - Deployment sequence control flow chart 21](#_Toc10465424)

[Figure 5 deployment sequence flow chart 22](#_Toc10465425)

# Introduction

The purpose of this document is to explain, in English, how each system that is part of the satellite works. It contains flow charts to describe how the systems are executed along with extra detail to describe processes.

## Authors note

Creating this satellite program has been one of the most interesting challenges I have ever encountered, the systems created here are like ones you would only dream of creating in a game. I’m very thankful I’ve had the opportunity to solve the challenges here and create a truly awesome program.

* Richard Bamford

# Documentation references

Here you will find a reference to the specific chapter which fulfil the given Functional or Test specification entry.

## Main Arduino Program

* Functional Specification ID(s): SYSFS1, SYSFS2, SYSFS6, SYSFS11, COMMFS6, COMMFS9, COMMFS11, COMMFS18, DEBUGFS9
* Test Specification ID(s): MAINPROGT1, MAINPROGT2, MAINPROGT3, MAINPROGT4, MAINPROGT5, MAINPROGT5, MAINPROGT6, MAINPROGT8, MAINPROGT9, MAINPROGT10, MAINPROGT11, DEPLOYT3, AUTOINTT2, COMMST6, COMMST13

## Communications System

* Functional Specification ID(s): SYSFS8, SYSFS9, SYSFS10, COMMFS1, COMMFS2, COMMFS3, COMMFS4, COMMFS5, COMMFS6, COMMFS7, COMMFS9, COMMFS10, COMMFS11, COMMFS12, COMMFS13, COMMFS14, COMMFS15, COMMFS16, COMMFS17, COMMSFS22, COMMSFS23
* Test Specification ID(s): MAINPROGT5, MAINPROGT6, DEPLOYT5, DEPLOYT6, POWCONT2, HARDINT5, HARDINT6, HARDINT7, HARDINT8, HARDINT9, HARDINT11, COMMST1, COMMST2, COMMST3, COMMST4, COMMST5, COMMST6, COMMST7, COMMST8, COMMST9, COMMST10, COMMST11, COMMST12, COMMST13, COMMST14, COMMST15, COMMST1

## Safety & Security System

* Functional Specification ID(s): SYSFS3, SYSFS4, COMMFS6, COMMFS7
* Test Specification ID(s): MAINPROGT3, SAFESECT1, SAFESECT2, SAFESECT3

## Deployment System

* Functional Specification ID(s): SYSFS2, COMMFS9, DEBUGFS9, COMMFS13, COMMFS17, COMMFS10
* Test Specification ID(s): DEPLOYT1, DEPLOYT2, DEPLOYT3, DEPLOYT4, DEPLOYT5, DEPLOYT6, DEPLOYT7, COMMST6, COMMST7, COMMST11

## Battery Conservation System/Automatic Interval Control System

* Functional Specification ID(s): SYSFS6, SYSFS12
* Test Specification ID(s): AUTOINTT1, AUTOINTT2, AUTOINTT3

## Battery Temperature Control System

* Functional Specification ID(s): SYSFS7
* Test Specification ID(s): POWCONT1, POWCONT2

## Configuration System

* Functional Specification ID(s): SYSFS1, SYSFS3
* Test Specification ID(s):

## Hardware Interface System

* Functional Specification ID(s): SYSFS5
* Test Specification ID(s): HARDINT1, HARDINT2, HARDINT3, HARDINT4, HARDINT5, HARDINT6, HARDINT7, HARDINT8, HARDINT9, HARDINT10, HARDINT11

## System Information System

* Functional Specification ID(s): COMMFS3, COMMFS6
* Test Specification ID(s): COMMST4

# Main Arduino Program

## Background

The Arduino platform operates on an ATMEGA328p chip, which contains 2KB of RAM and 32KB of Flash memory. This platform provides a C++ IDE that is used to program the chip, the IDE exposes the *Setup* and *Loop* functions to execute user code. (Arduino Loop Function Reference, 2019).

## Setup Function

### Description

The setup function is a built-in function provided by the Arduino platform. It is run once, every time the Arduino is powered on. Global variables are initialized when the program begins and can be configured/initialized here, in the setup function.

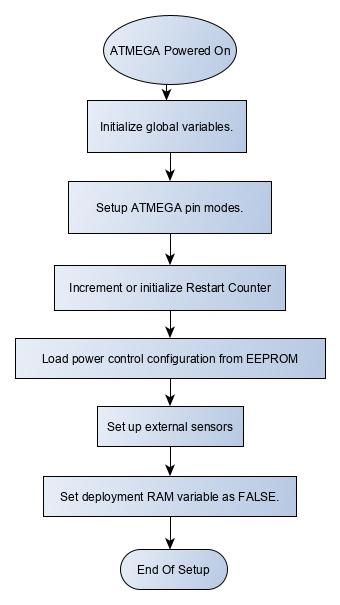


Figure 1 - Arduino sketch setup flow chart

## Looping function

### Description

The Loop function provided by the Arduino platform is run repeatedly for as long as the Arduino is powered on. It begins its execution after the Setup function has completed.

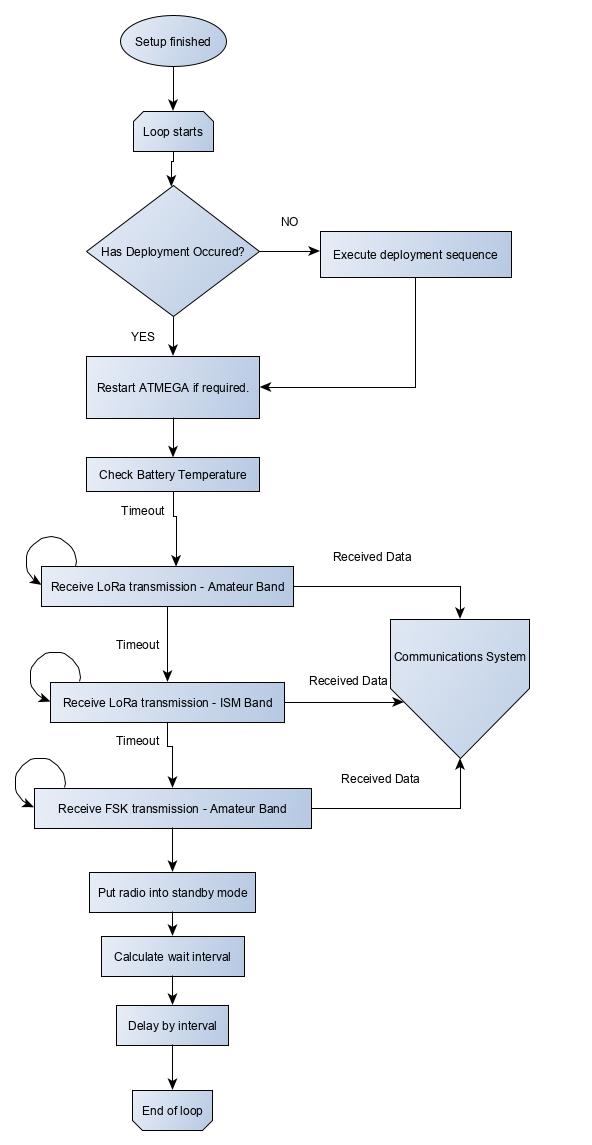


Figure 2 - Arduino sketch loop function

# Communication System

## Background

Using an SX1278 LoRa radio module and an omnidirectional antenna the satellite transmits, receives and processes radio signals.

The RadioLib Arduino library is used to interface with the SX1278 radio module. There are several restrictions with the radio communications library RadioLib (Gromeš, 2019):

1. The transmission and reception of radio signals only works in the main loop of the Arduino program.
2. The receive and transmission cycles are quantized. Therefore, the communication between satellite and ground stations is not 100% reliable.
3. Only one mode of radio communication is supported at one time e.g. Listen LORA for 10 seconds, Listen LORA ISM for 10 seconds, listen FSK for 10 seconds and finally transmit LORA.

This system must accommodate for these restrictions and provide an expected interface.

## Architecture

The communications system functions by implementing a flag based IF-ELSE branching block. A list of booleans are defined as global variables which indicate which functions to execute. This way a message can be sent from anywhere in the program by raising a flag.

## Program flow

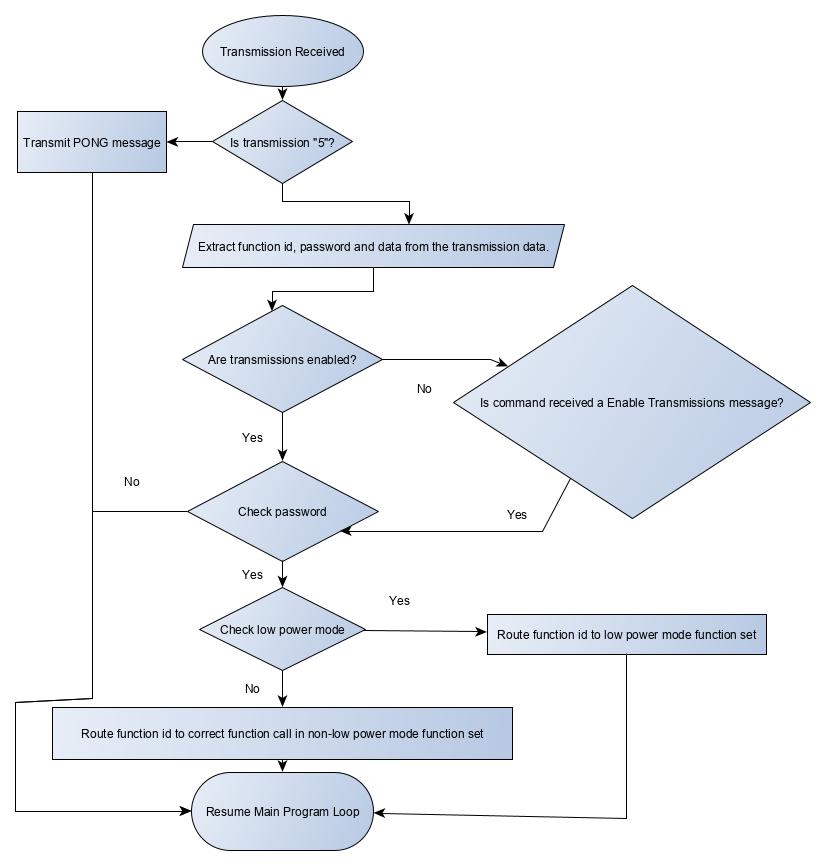


Figure 3 - Communications system flow chart

## Flag Definitions

|  |  |
| --- | --- |
| Flag | Description |
| Started | Raised to TRUE when the Arduino first starts up. |
| Stopped | Raised to TRUE when the Arduino is going to shutdown. |
| Deployment Success | Raised to TRUE when the deployment has succeeded. |
| Ping | Raised to TRUE when the PING transmission is receives. |
| Transmit RTTY Callsign | Raised to TRUE every 10 loop cycles to transmit the RTTY callsign. |
| Stop | Raised to TRUE to prevent the watchdog heartbeat. |
| Transmit Repeater Message | Raised to TRUE to transmit the current stored repeater message. |

## Protocol Definition

### Protocol Description

The protocol is how the transmissions should be formed for the satellite to understand them. Each character in the string (see column `Format` in table TODO) is an ASCII encoded-2 byte data type. Each string must end with a NULL character.

The protocol format is as follows, where the `;` character defines the start of the data section.

**{PASSWORD}{FUNCTION ID};{DATA}**

### Protocol Lookup Table

|  |  |  |  |
| --- | --- | --- | --- |
| Format | Function ID | Direction | Description |
| FOSSASAT-11; | 1 | To Ground | Powered on. |
| FOSSASAT-12; | 2 | To Ground | Powering off. |
| FOSSASAT-14; | 4 | To Ground | Deployment successful. |
| FOSSASAT-15; | 5 | To Satellite | PING command. |
| FOSSASAT-16; | 6 | To Ground | PONG command. |
| FOSSASAT-17; | 7 | To Satellite | Disable transmitting. |
| FOSSASAT-18; | 8 | To Satellite | Enable transmitting. |
| FOSSASAT-19;1,0.00,0.00,0.00,0.00,1,12,0.00 | 9 | To Ground | System Information.  Low Power Mode 1/0, Battery V, Solar Cell A V, Solar Cell B V, Solar Cell C V, Deployment State, Restart Counter, Board Temperature. |
| FOSSASAT-110; | 10 | To Satellite | Zero EEPROM. |
| FOSSASAT-111; | 11 | To Satellite | Run deployment sequence. |
| FOSSASAT-113;New Callsign | 13 | To Satellite | Set callsign. |
| FOSSASAT-114; | 14 | To Satellite | Restart. |
| FOSSASAT-115;Repeater message | 15 | To Satellite | Re-transmit given data. |
| FOSSASAT-116;Repeated Message | 16 | To Ground | Transmitted repeater message. |
| FOSSASAT-117; | 17 | To Satellite | Transmit the callsign using spreading factor 7. |
| FOSSASAT-118; | 18 | To Ground | Transmitted with SF7 for doppler debugging. |
| FOSSASAT-119; | 19 | To Satellite | Switch satellite to alternative spreading factor. |
| FOSSASAT-120; | 20 | To Satellite | Switch the satellite to the normal spreading factor. |
| FOSSASAT-121; | 21 | To Satellite | Enable MPPT charging circuit temperature switching. |
| FOSSASAT-122; | 22 | To Satellite | Disable MPPT charging circuit temperature switching. |
| FOSSASAT-123; | 23 | To Satellite | Enable low power mode system. |
| FOSSASAT-124; | 24 | To Satellite | Disable low power mode system. |
| FOSSASAT-125; | 25 | To Satellite | Manually transmit single system information packet. |
| FOSSASAT-126; | 26 | To Satellite | Turn MPPT keep-alive mode on. |
| FOSSASAT-127; | 27 | To Satellite | Turn MPPT normal mode. |

# Safety & Security System

## Background

The safety & security system exposes several functions that are used throughout the program to check error codes, strings and deployment EEPROM values.

## RadioLib (SX1278) error codes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Context/Function (RadioLib) | Level of Severity (HIGH, MED, LOW, NONE) | Error Code (integer enums) | System Response | Reason |
| “.Begin()” / initialization function. | NONE | ERR\_NONE | None. | Indicates the lora library has initialized the LORA object properly, without errors. |
|  | HIGH | ERR\_CHIP\_NOT\_FOUND | None. | No chip found is a critical error. |
|  | HIGH | NO VALUE | None. | Unknown value returned by LORA library, suggests program malfunction. |
| “.Transmit()” / transmit a message. | LOW | ERR\_PACKET\_TOO\_LONG | None. | Transmission with too much data, carry on as normal. |
|  | HIGH | ERR\_TX\_TIMEOUT | None. | Transmissions should never time out. This suggests SX1278 failure. If we cannot transmit the satellite is silent. |
|  | NONE | ERR\_NONE | None. | Transmission sent correctly. |
|  | LOW | NO VALUE | None. | We do not know whether the transmission was sent or not. The software assumes that it was sent, so that another system can catch it as an error/manual restart. |
| “.Receive()” / Receive a message | LOW | ERR\_RX\_TIMEOUT | None. | This is a normal operating error code when no transmissions are received. |
|  | LOW | ERR\_CRC\_MISMATCH | None. | This code should not be reported. If it is then we are receiving faulty transmissions. Restart to zero. |
|  | NONE | ERR\_NONE | None. | This is the code that returns when the function receives data. |
| Setting configuration function | HIGH | ERR\_INVALID\_BANDWIDTH | None. | All of the settings are hard coded in the satellite. If we are reading this error then the memory has been corrupted. |
|  | HIGH | ERR\_INVALID\_SPREADING\_FACTOR | None. | “ |
|  | HIGH | ERR\_INVALID\_CODING\_RATE | None. | “ |
|  | HIGH | ERR\_INVALID\_FREQUENCY | None. | “ |
|  | HIGH | ERR\_INVALID\_OUTPUT\_POWER | None. | “ |

## EEPROM deployment fault checking.

### introduction

The deployment state is saved to the EEPROM. On deployment we check this value, this system samples before and after the deployment sequence has executed and checks for EEPROM faults.

### EEPROM fault modes

|  |  |  |  |
| --- | --- | --- | --- |
| Pre-Sequence Value | Post-Sequence Value | Meaning | Action |
| 0 | 0 | No change in EEPROM address, deployment tried to happen but failed.  Failed to allocate to eeprom. | Zero the EEPROM. |
| 0 | 1 | First deployment. | No action. |
| 1 | 1 | Manual deploy executed. | No action. |
| 1 | 0 | EEPROM fault | Zero the EEPROM. |

# Deployment System

## Background

The deployment system controls when the antenna and solar panels are released. It has three different settings in which to run; the debugging mode, the integration mode and the real launch setting.

## Sequence Control

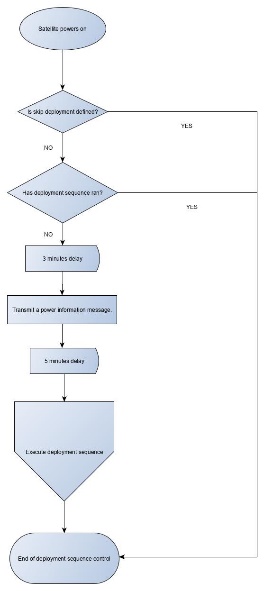


Figure 4 - Deployment sequence control flow chart

## Debugging mode

The deployment sequence is skipped to perform debug cycles quicker.

## Integration mode

At integration, the satellite is tested by powering it on, waiting 3 minutes and receiving a system information packet. Then a 5-minute delay allows the integration engineer to power the satellite off before the deployment sequence is executed.

## Launch mode

At launch the operating mode is the same as the integration mode however, the satellite can power on after the 5 minutes has passed.

## Deployment Sequence

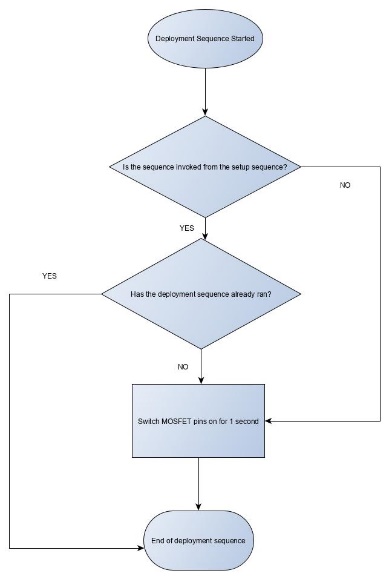


Figure 5 deployment sequence flow chart

# Battery Conservation System/Automatic Interval Controller

## Background

The main Arduino loop function has a delay between each execution. The satellite controls this delay so that when on low battery the delay is for a longer amount of time, this allows for the battery to charge with no load for longer when it is running low.

## Low Power mode

If the battery drops below 3.3V the satellite switches to a “low power mode”. In this mode the satellite will only reply to manual commands to send a system information packet. Additionally, the command to disable/enabled the low power mode will be exposed in case the satellite gets stuck in that mode.

## Configuration saved to EEPROM.

The Low Power Mode, MPPT temperature switching and MPPT enable/disable flags are saved to the EEPROM and loaded during restarts.

# Battery Temperature Control System

## Background

The Li-Po battery cannot be charged when the temperature is below 0 degrees, therefore we read the battery temperature sensor and if it’s less than 0, we turn the MPPT charging circuit off. Turning the MPPT charging circuit off disabled the solar panels from charging it.

# Appendix A – Diagrams

## Arduino sketch setup function flowchart

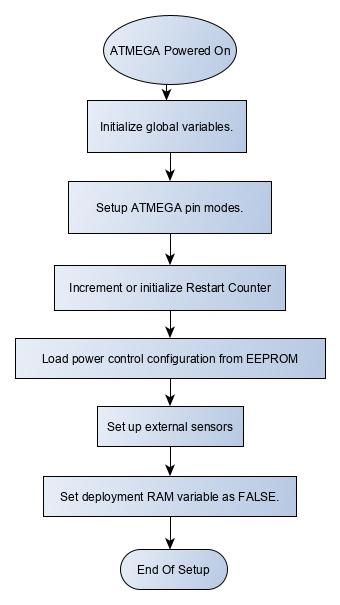


Figure 1 - Arduino sketch setup flow chart

## Arduino loop function flow chart

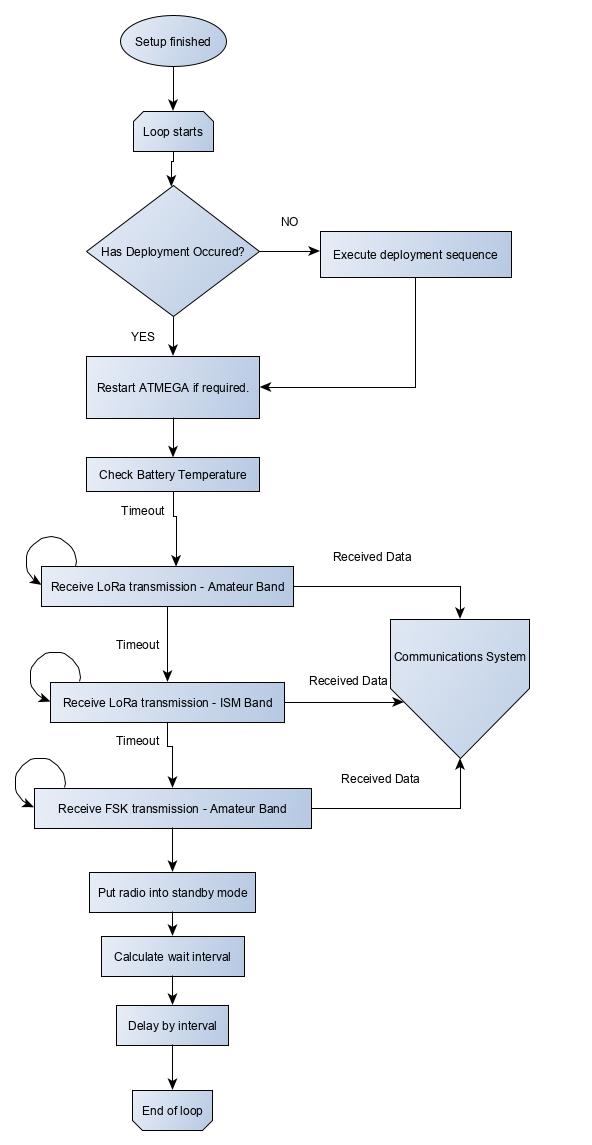
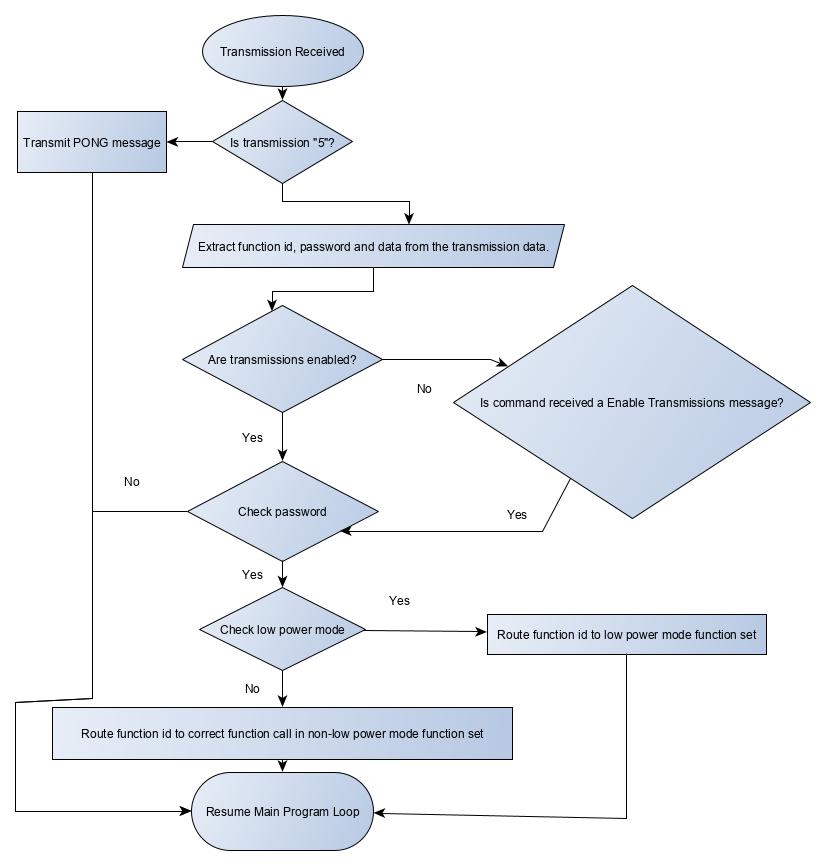


Figure 2 - Arduino sketch loop function

## Communication system program flow chart

Figure 3 - Communications system flow chart

## Deployment sequence control flow chart

Figure 4 - Deployment sequence control flow chart

## Deployment sequence flow chart

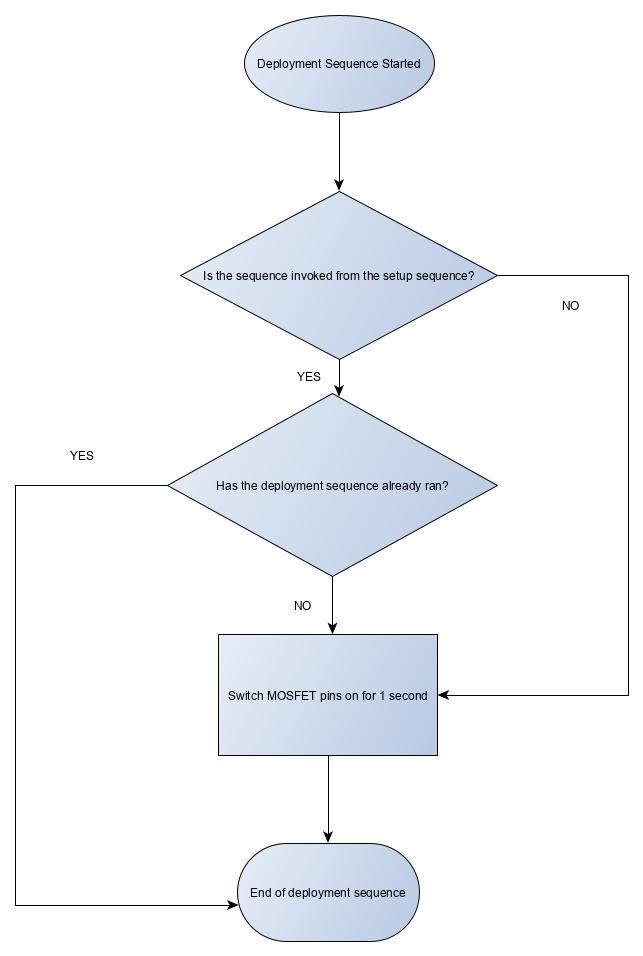


Figure 5 deployment sequence flow chart

# References

*Arduino Loop Function Reference*. (2019, May 17). Retrieved from Arduino: https://www.arduino.cc/en/Reference/Loop?setlang=it

Gromeš, J. (2019, 05 05). *RadioLib Git Repository*. Retrieved from Github: https://github.com/jgromes/RadioLib