

Installation Manual

Group: I-TT-4N2, I-IT-4N2

Version	Date	Author	Change
1.0	12.12.2017	Apurva Ganoo	-
1.1	15.12.2017	Apurva Ganoo	Additions made

Table of Contents

- Introduction 3
- Functions 4
 - IR Sensor 5
 - Change Resolution..... 5
 - Get Humidity 6
 - Get Temperature 8
 - Main..... 8
 - Read Byte 9
 - Send Byte..... 10
 - Send Start..... 11
 - Write Register..... 12
- Network Configurations 13
- Hardware 16
 - Casing..... 16
 - Infrared Sensor 16
 - Infrared Sensor **Error! Bookmark not defined.**
 - Infrared Sensor **Error! Bookmark not defined.**

Introduction

This technical specification is designed to provide a concise overview of the City Atmospheric Measurement System (CAMS) Project that was undertaken by the Information Technology students from the Finnish and English side at the Vaasa University of Applied Sciences.

The goal of the project was to deliver a working prototype, or a node, which was capable of measuring at least 3 environmental variable using different sensors, which would then push data using the LoRaWAN network to The Things Network gateway which was hosted at our university.

This project, if undertaken, will require a basic understanding of electronics and electrical components, some programming knowledge; particularly in C, and an understanding of LoRaWAN technology and protocol along with knowledge about the Things Network.

The general system architecture can be seen below (see fig.1).

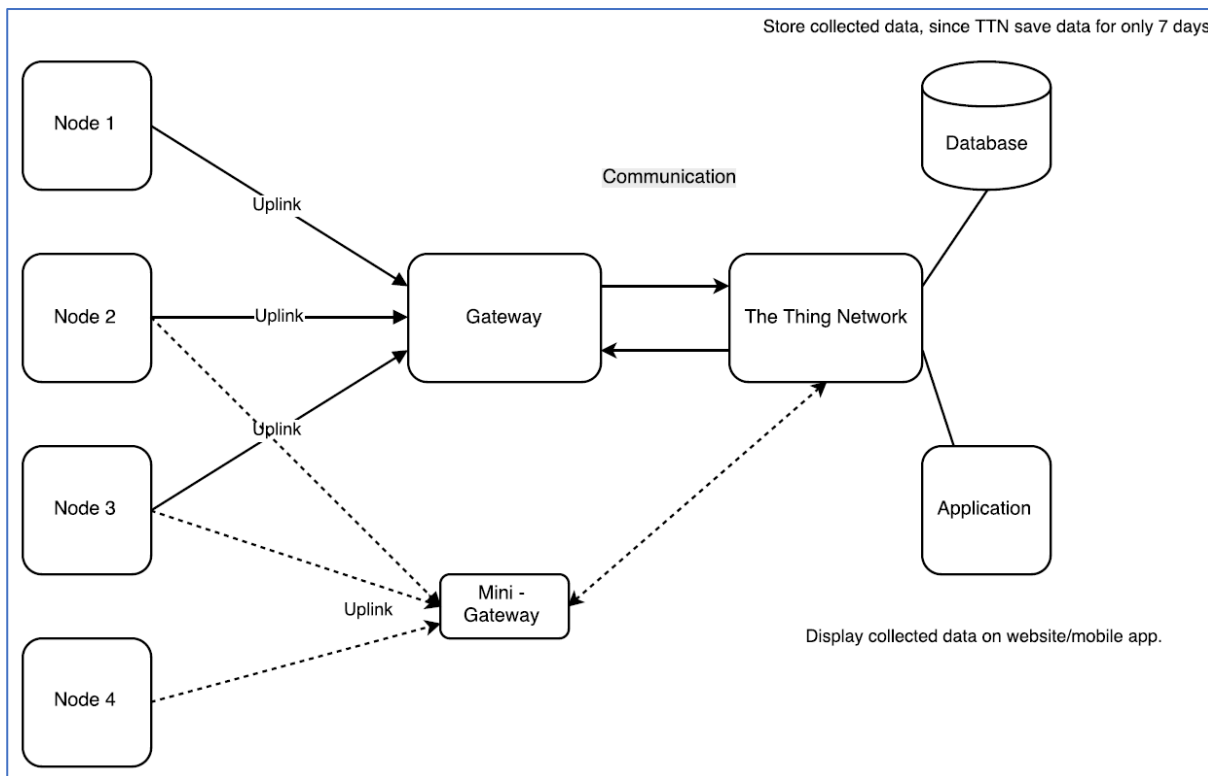


Fig.1 – System Architecture

Functions

Listed below are the functions, programmed by the Software teams, onto the microcontroller to help configure the sensors that we have chosen and to push the data, in the right format, to the LoRa Gateway (see fig.1).

<i>Function Name</i>	<i>Function Description</i>
IR Sensor	Check if Infrared sensor is blocked by something or not.
Change_Resolution	Change resolution of measurements
get_humi	Obtain humidity values from sensor
get_temp	Obtain temperature values from sensor
main	Saving obtained values of temperature and humidity
read_byte	Reading values
send_byte	Sending values forward
send_start	Sending initial signal to sensor to begin measuring

write_register	Write values to register on microcontroller
----------------	---

Fig.1 – Function Descriptions.

As such, we will now be going into details of each and individual functions; firstly, by providing relevant information, and secondly, by providing functional flowcharts to help further clarify the purpose and logic of each function.

IR Sensor

The IR sensor function is designed to help measure traffic data using an infrared sensor. This is done by checking whether the area between the sensor and the receiver has been blocked by any item, and if so, by how long. The functional flowchart can be seen in the diagram below (see fig.2). The function then creates a simple counter, which adds for every time the sensor and the receiver had been blocked by an external object.

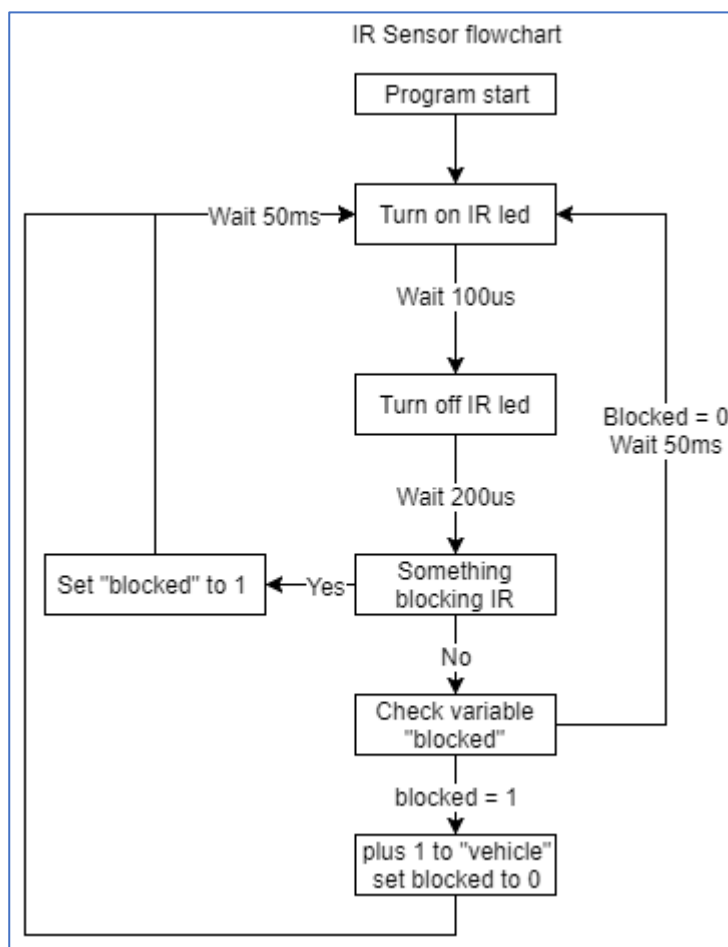


Fig.2 – Function Flowchart for IR Sensor

Change Resolution

The change resolution function was created to help configure the monitor sensor and choose between the two resolution options available on the temperature sensor (see fig.3).

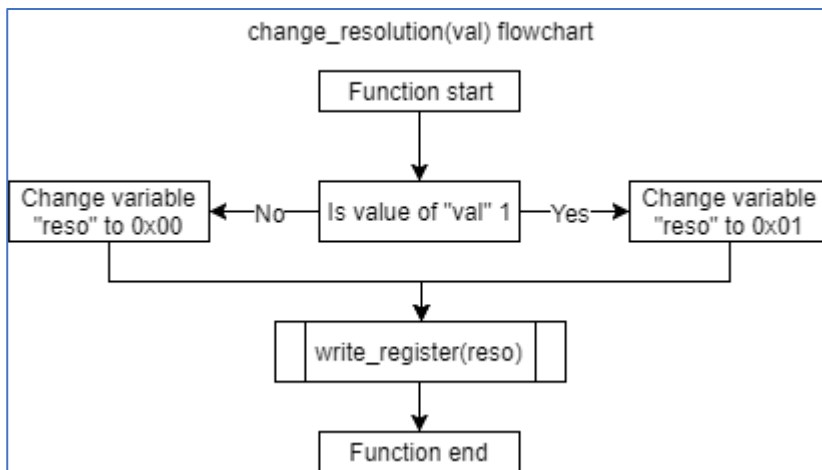


Fig.3 Function Flowchart for Change Resolution

Get Humidity

The get humidity function was designed to help read the measured humidity data from the temperature sensor. It is created to read the data in the right format, and with the right amount of precision.

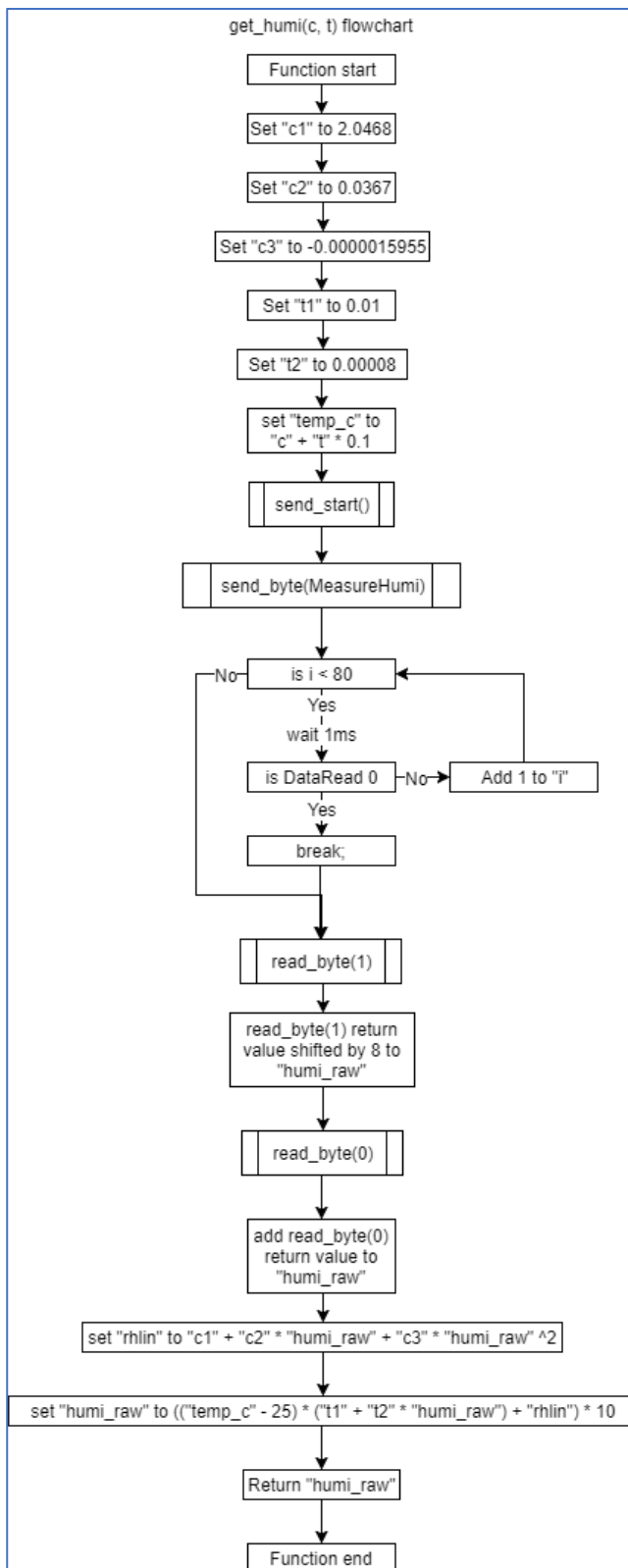


Fig.4 Function Flowchart for Get Humidity

Get Temperature

The get humidity function was designed to help read the measured temperature data from the temperature sensor. It is created to read the data in the right format, and with the right amount of precision.

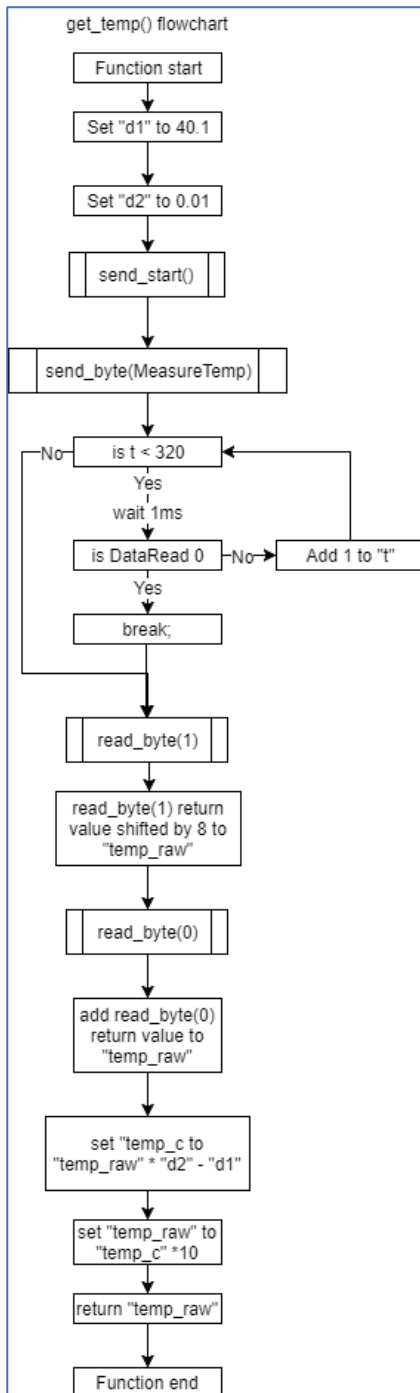


Fig.5 Function Flowchart for Get Temperature

Main

The main function's purpose is to save the obtained/measured values of the temperature and humidity data.

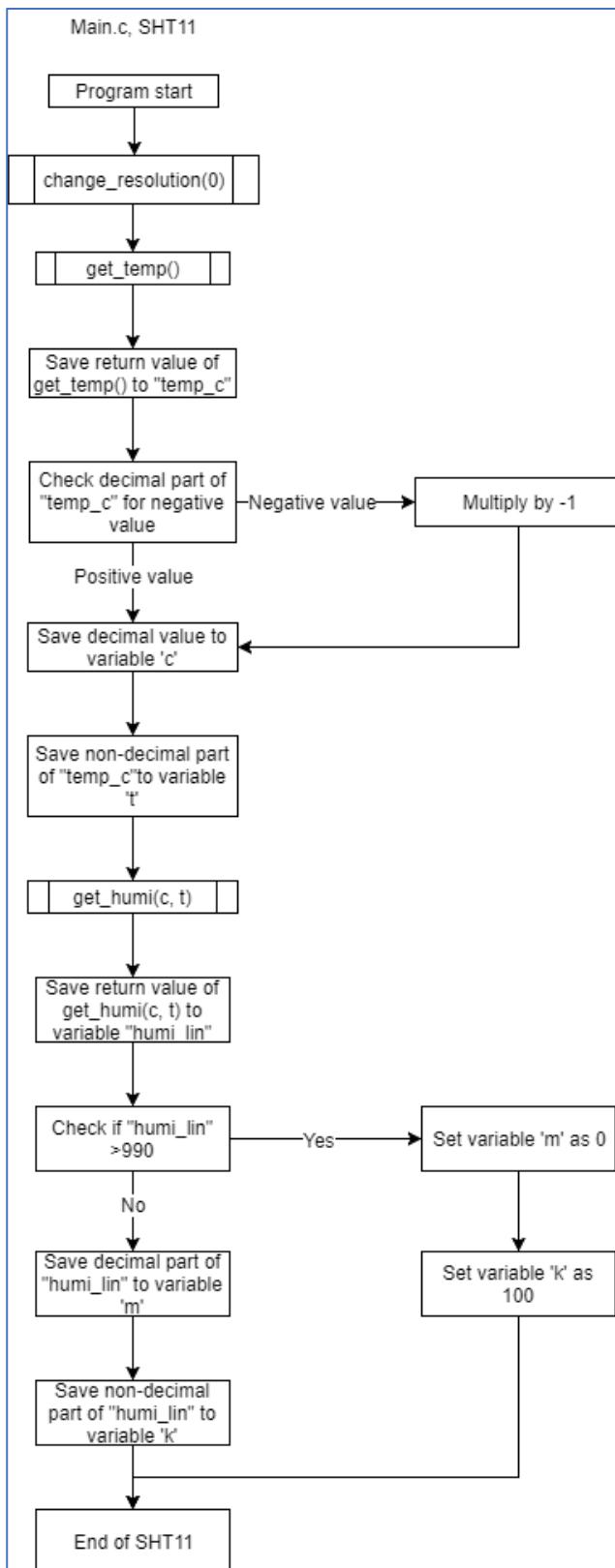


Fig.6 Function Flowchart for Main Function

Read Byte

The read byte function is designed to read values, and reassign the values of the read data. This function is for the IR sensor and to monitor its data.

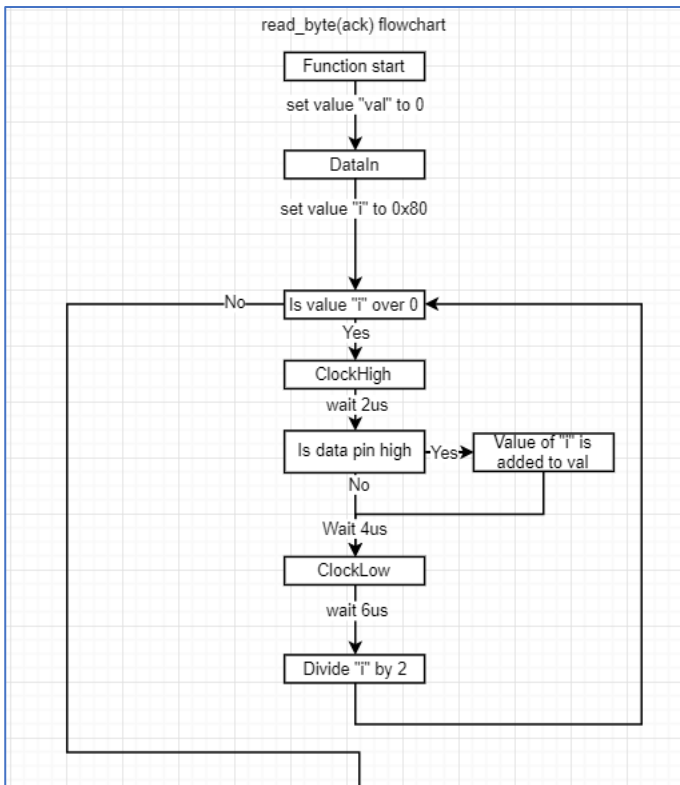


Fig.7 Function Flowchart for Read Byte Function

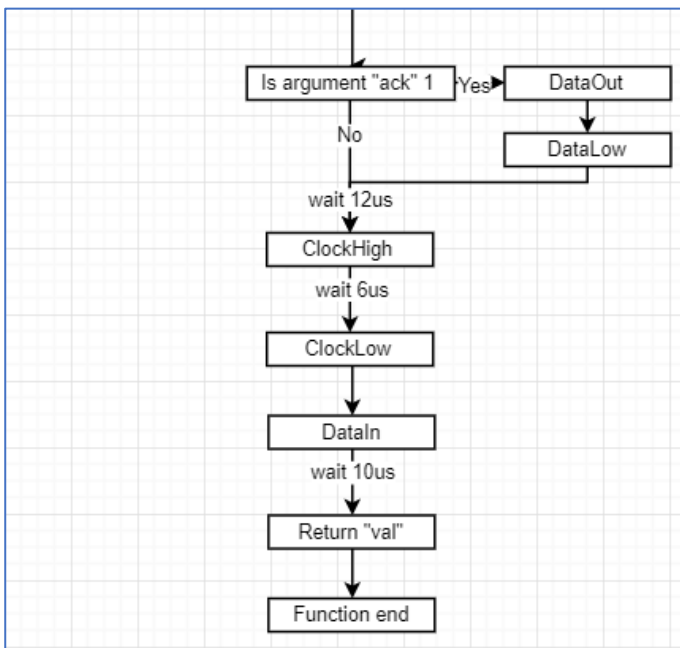


Fig.8 Function Flowchart for Read Byte Function continued

Send Byte

The send byte function is designed to send values, and reassign the values of the read data. This function is for the IR sensor and to monitor its data.

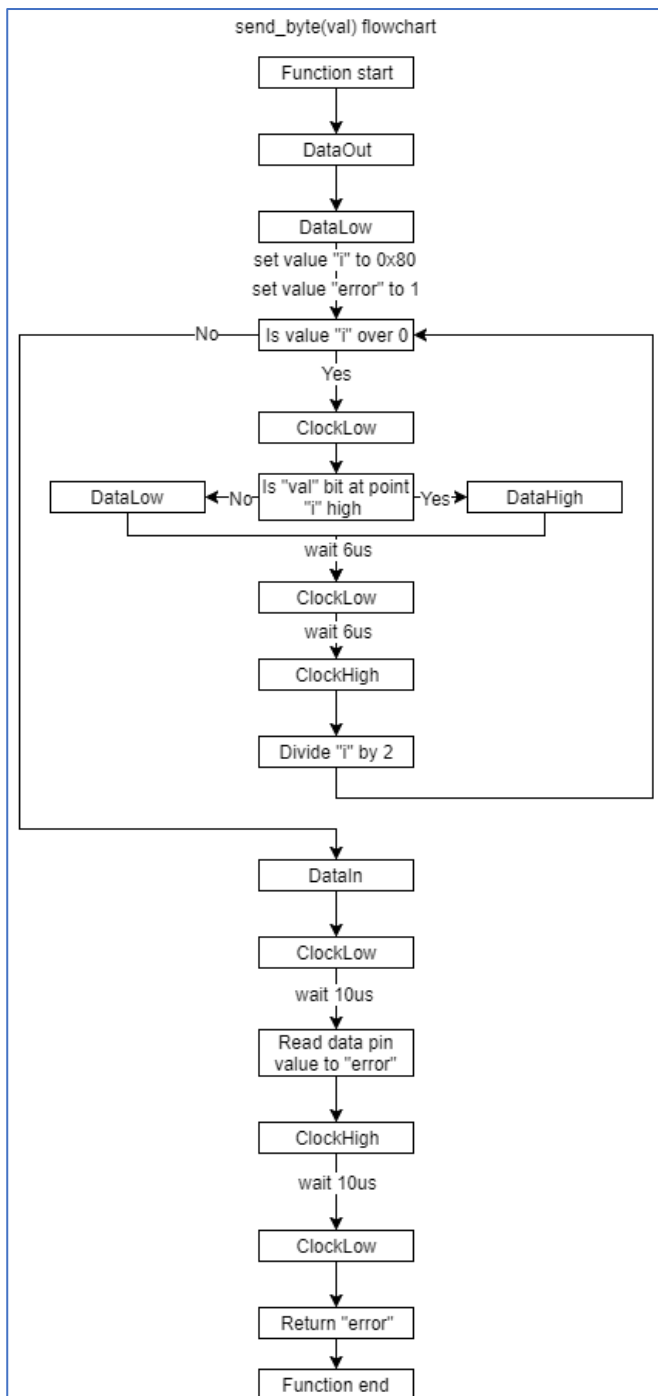


Fig.9 Function Flowchart for Send Byte

Send Start

The send start function is designed to send an initiate command to the sensor to begin recording using the IR sensor and to send out pulses using the IR transmitter.

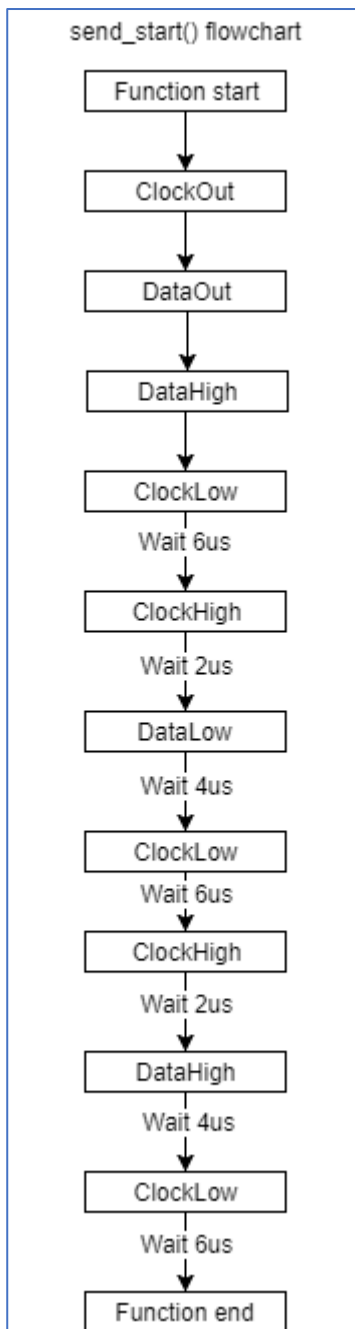


Fig.10 Function Flowchart for Send Start

Write Register

The write register function is designed to write the values obtained from the sensor data to a register to store these values.

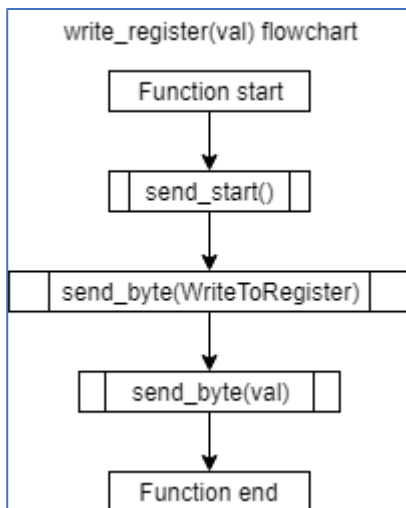
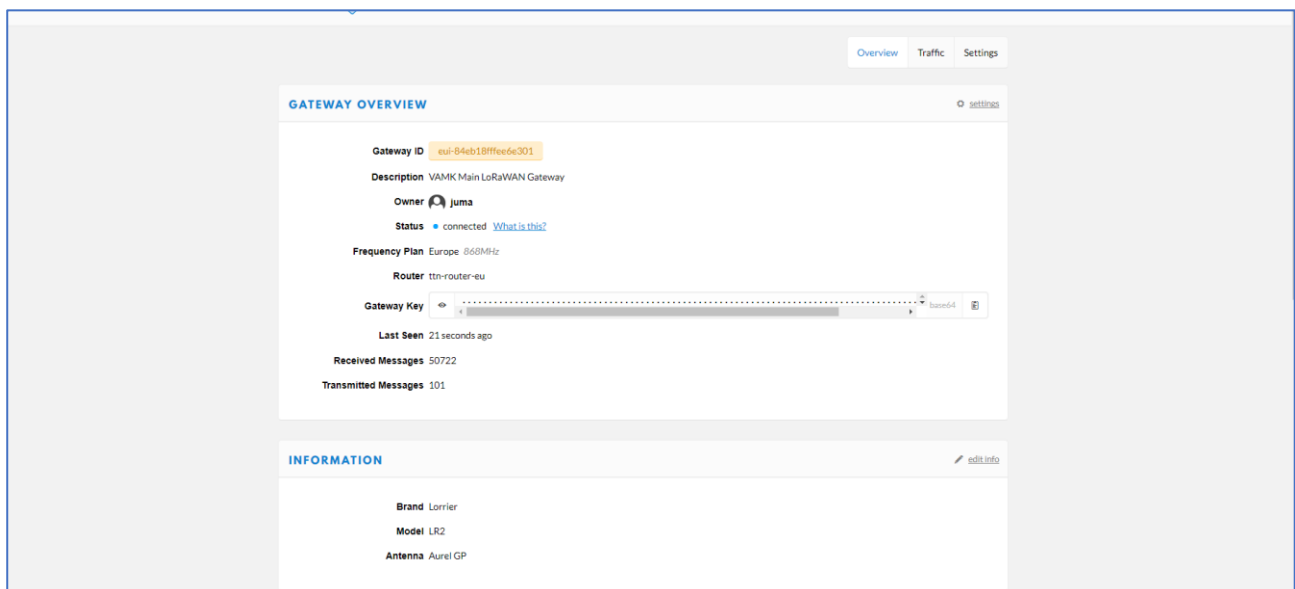


Fig.11 Function Flowchart for Write Register

Network Configurations

The network configuration for this project involves two parts: the node itself, and The Things Network (TTN). The goal of the project is to configure the nodes in a way that they send, periodically, and automatically their sensor data onto the Things Network by sending the data to the right gateway. As such, the first part involved choosing the right LoRa chip and configuring it to send data to the correct LoRa Gateway which was hosted at a central location. This involved configuration from both, the sending, and the receiving site. As such, the gateway work status can be seen in the figure below (see fig.12).



This (fig.12) provides necessary information regarding the gateway that needs to be considered when configuring the nodes. It also contains status updates that can be used to check whether the messages are being sent as intended, and whether the gateway is live. The gateway configurations can be accessed through the official TTN website and logging in with one's credentials (see fig.13 and 14).

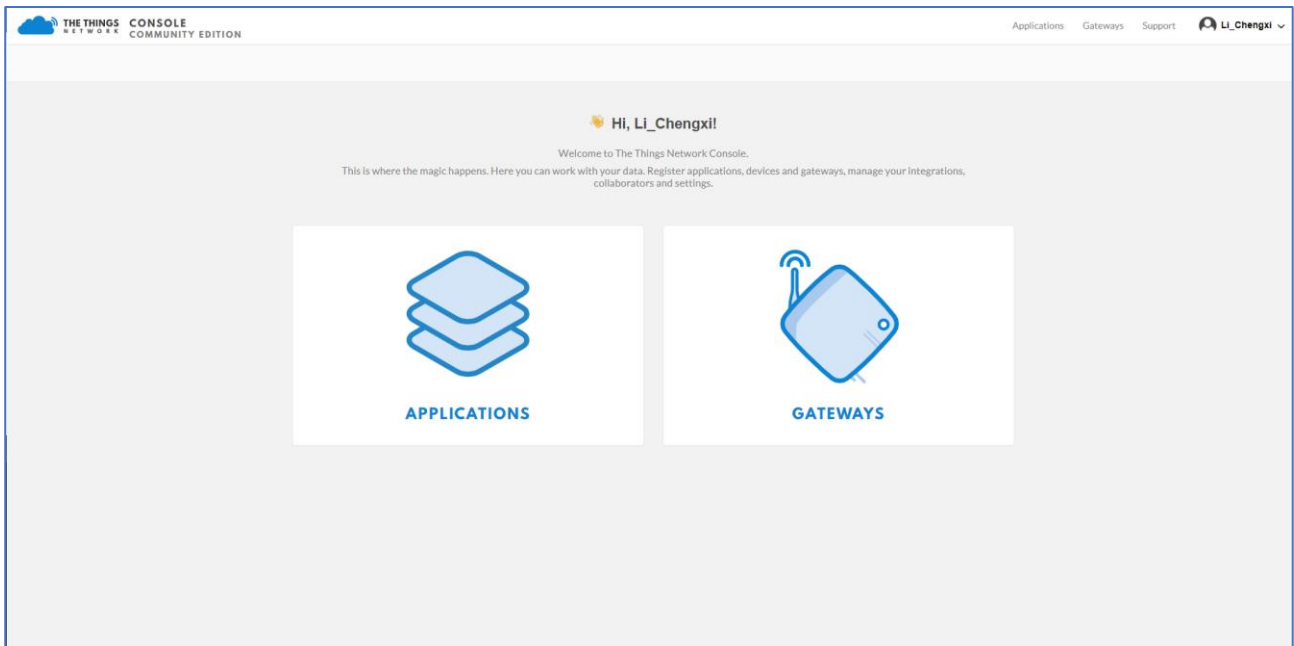


Fig.13 – Home/Dashboard screen of TTN Webpage

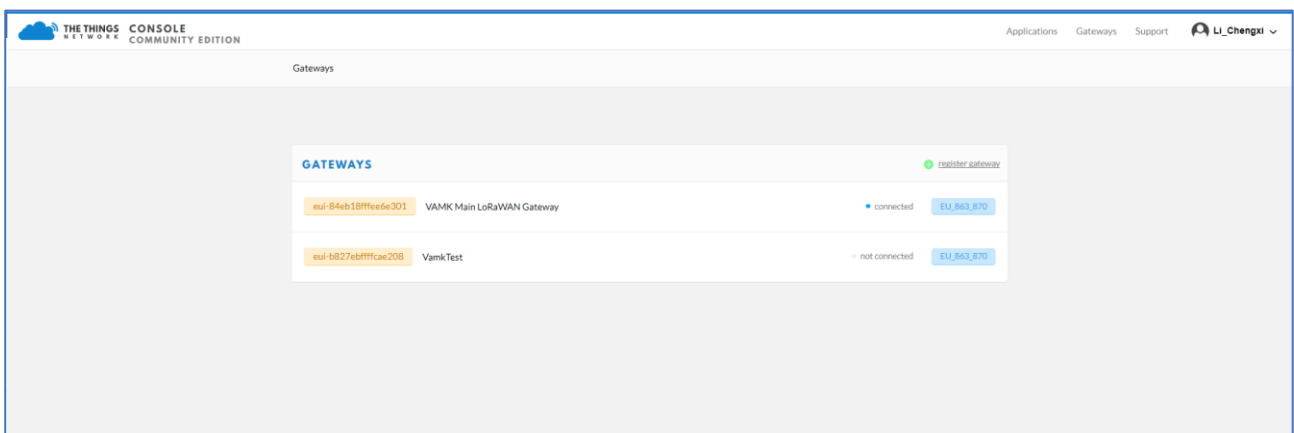


Fig.14 – Information regarding configured gateways on TTN. VAMK MainLoRaWAN Gateway is the one to be used for this project.


Once the configurations had been completed correctly, we could verify our results by sending data, at timed intervals, to the TTN Gateway from our node itself.

From the network side, we also had to ensure that we stored our data at some secure location as TTN can only store data for a maximum period of 7 days. As such, we had to create and store the data in a local database. For the basic integration, we used the default data storage capabilities provided by TNN (see fig.15).

INTEGRATION OVERVIEW

Status ● Running

Integration info [go to platform](#)

Platform  Data Storage (v2.0.1)

Author The Things Industries B.V.

Description Stores data and makes it available through an API. Your data is stored for seven days.

Fig.15 – Database Integration from TTN

However, we then had to create a system to fetch the data from TTN in JSON format and then save the data to CSV format to ensure that our data would not be lost after 7 days (see fig.16).

```

headers = {
    'Accept': 'application/json',
    'Authorization': 'key ttn-account-v2.W9J8E46eTdFCF5R60TPPZ0Z8daPSws-U6lXbfIRz1Hk',
}

r=requests.get('https://technobothnia.data.thethingsnetwork.org/api/v2/query', headers=headers)

```

Fig.16 – Fetching data and storing it to CSV format

The CSV data, a sample of which can be seen below, was then pushed to a local database (see fig.17):

,device_id,raw,time

0,01,SGVsbG8gd29ybGQhIFswXQ==,2017-11-09T18:57:55.050134648Z

1,01,SGVsbG8gd29ybGQhIFswXQ==,2017-11-09T19:02:55.054049853Z

2,01,SGVsbG8gd29ybGQhIFswXQ==,2017-11-09T19:07:55.060989575Z

3,01,SGVsbG8gd29ybGQhIFswXQ==,2017-11-09T19:12:55.071874751Z

```

Database changed
mysql> show columns from ttn1;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| index      | bigint(20) | YES  | MUL | NULL    |       |
| device_id  | text      | YES  |     | NULL    |       |
| raw        | text      | YES  |     | NULL    |       |
| time       | text      | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

```

```
mysql> select * from ttn1;
```

index	device_id	raw	time
0	01	SGVsbG8gd29ybGQhIFswXQ==	2017-11-09T18:17:54.987721855Z
1	01	SGVsbG8gd29ybGQhIFswXQ==	2017-11-09T18:22:54.987306125Z
2	01	SGVsbG8gd29ybGQhIFswXQ==	2017-11-09T18:27:54.999369828Z
3	01	SGVsbG8gd29ybGQhIFswXQ==	2017-11-09T18:32:55.045123752Z
4	01	SGVsbG8gd29ybGQhIFswXQ==	2017-11-09T18:37:55.014951767Z
5	01	SGVsbG8gd29ybGQhIFswXQ==	2017-11-09T18:42:55.02533783Z
6	01	SGVsbG8gd29ybGQhIFswXQ==	2017-11-09T18:47:55.029390823Z
7	01	SGVsbG8gd29ybGQhIFswXQ==	2017-11-09T18:52:55.046802913Z
8	01	SGVsbG8gd29ybGQhIFswXQ==	2017-11-09T18:57:55.050134648Z
9	01	SGVsbG8gd29ybGQhIFswXQ==	2017-11-09T19:02:55.054049853Z
10	01	SGVsbG8gd29ybGQhIFswXQ==	2017-11-09T19:07:55.060989575Z
11	01	SGVsbG8gd29ybGQhIFswXQ==	2017-11-09T19:12:55.071874751Z

Fig.17 – Data stored into a local database

Hardware

The hardware part consisted of developing the PCBs, the relevant schematics, the choosing of the right components, and developing the overall physical architecture of the nodes.

Casing

The casing refers to the hardware casing that should be used to store all the electronic components, thereby protecting them from external environmental factors. The casing was designed to be minimalistic yet sufficient and with the possibility of adding in further components and sensors, with minimal effort (see fig. 17, 18, 19).

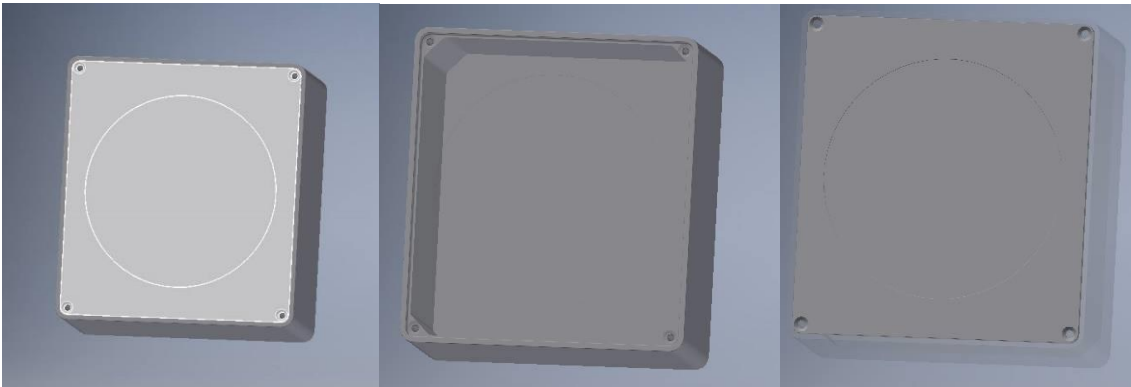


Fig.17 – Casing, Fig.18 – Casing Interiors, Fig.19 – Casing Continued (Left to Right)

Infrared Sensor

The schematics for the infrared sensor, which shall be used to monitor the traffic present in an area, can be seen below (see fig.20 and 21). As such, the schematic has been divided into 2 major parts: the receiver module, and the transmitter module.

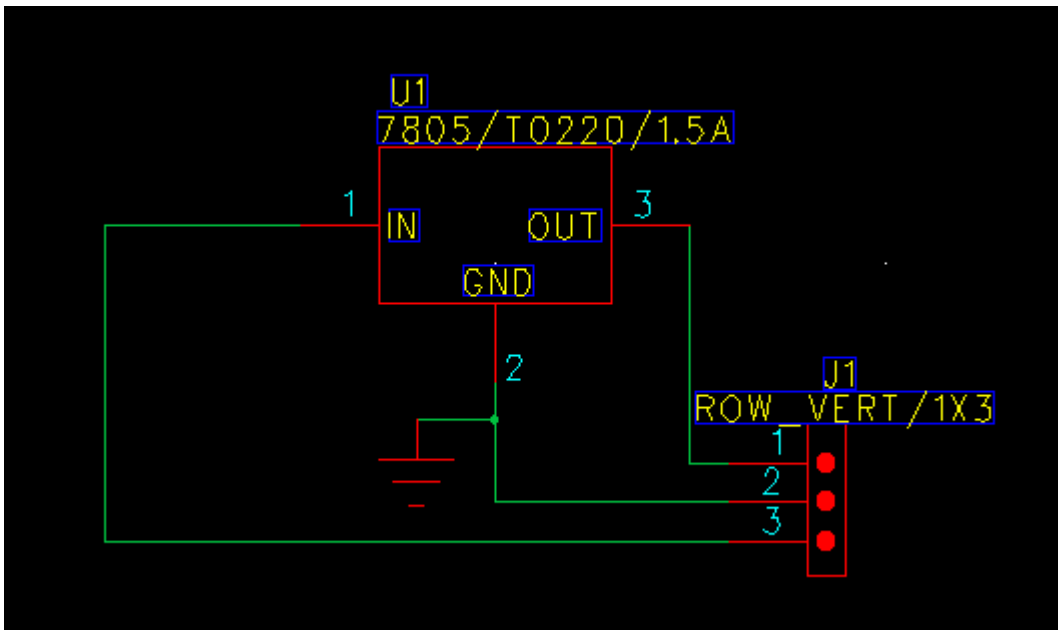


Fig.20 – Receiver module of the infrared sensor

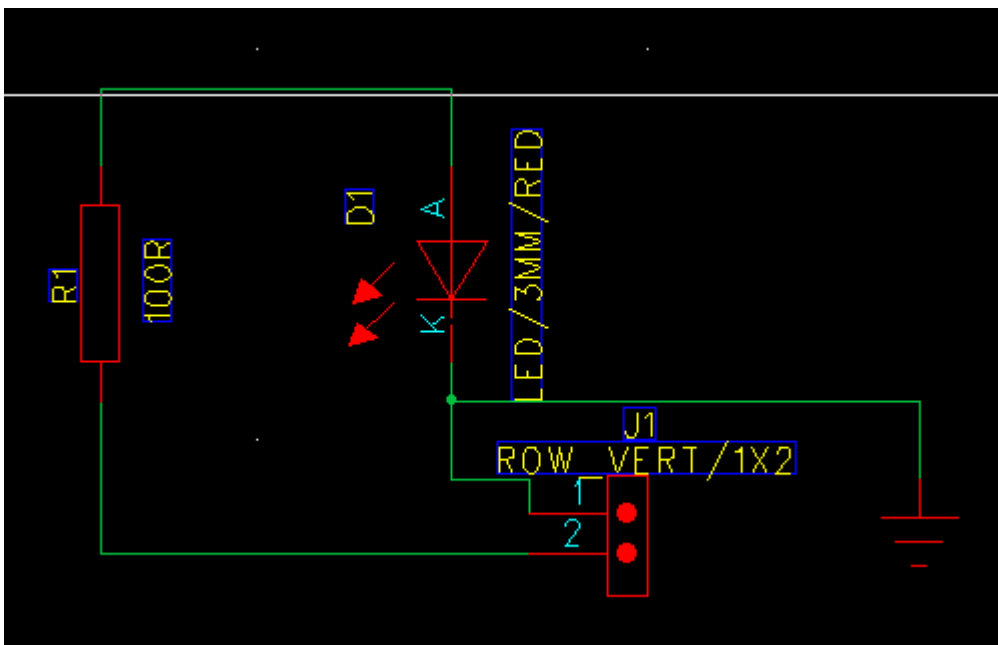


Fig.21 – Transmitter module of the infrared sensor

Pressure Based Sensor

The pressure based board sensor was another idea debated to be used to measure the traffic in an area. As such, the design of the board can be seen below (see fig.22).

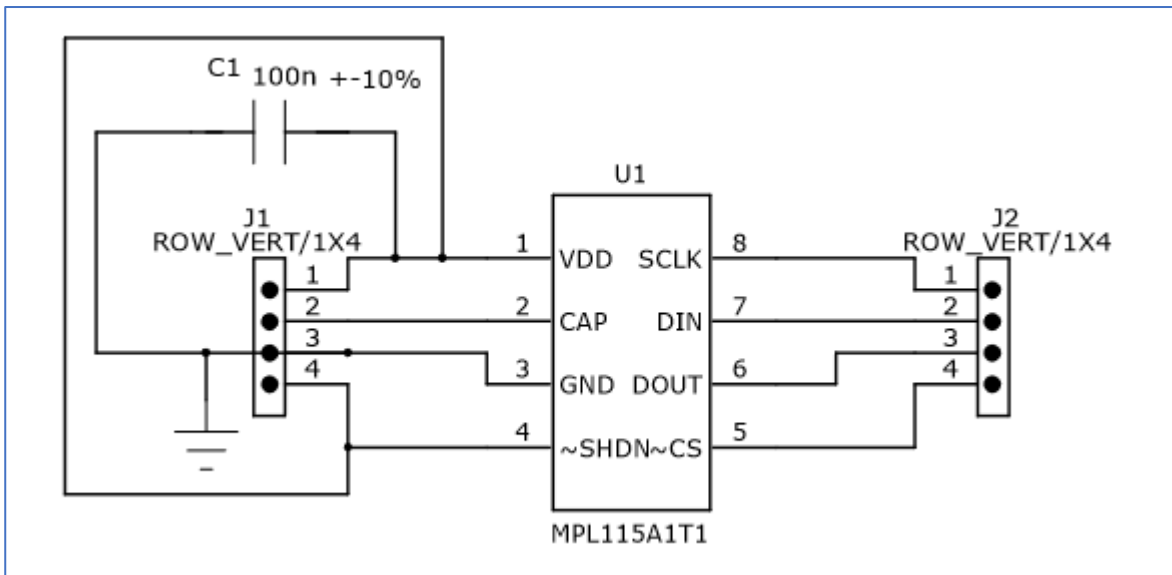


Fig.22 – Pressure Board Sensor

The idea with this design would be that pressure plates would be utilized to indicate whether a car/vehicle had crossed the desired or monitored road. As such, then it would be a simple counter function attached to find the total number of cars passing over the area over time.

Luminosity Sensor

The luminosity sensor was another sensor that was designed and considered to be added to the original node to increase the number of measurements. As such, the board's design can be seen below (see fig.23).

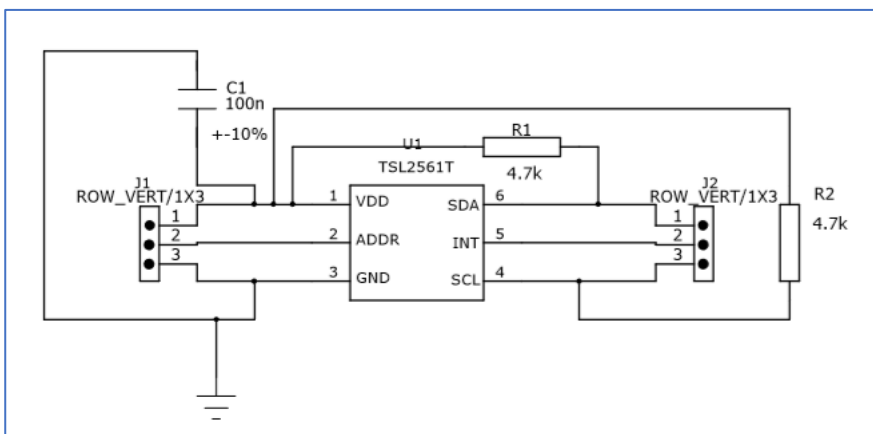


Fig.23 – Luminosity Board Sensor

Component/Order List

The mainboard facilitates the microcontroller and the LoRa chip which are to be used for our node. As such, the designs of the mainboard can be seen below (see fig.24, 25, 26).

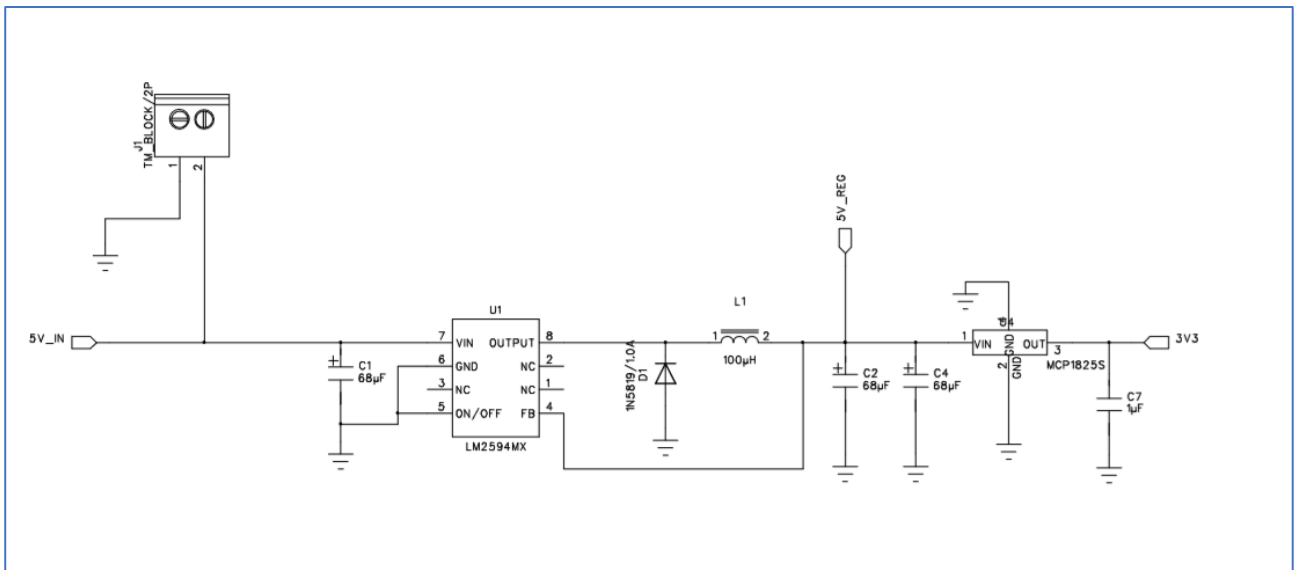


Fig.24 – Switch mechanism added to power on/off the node

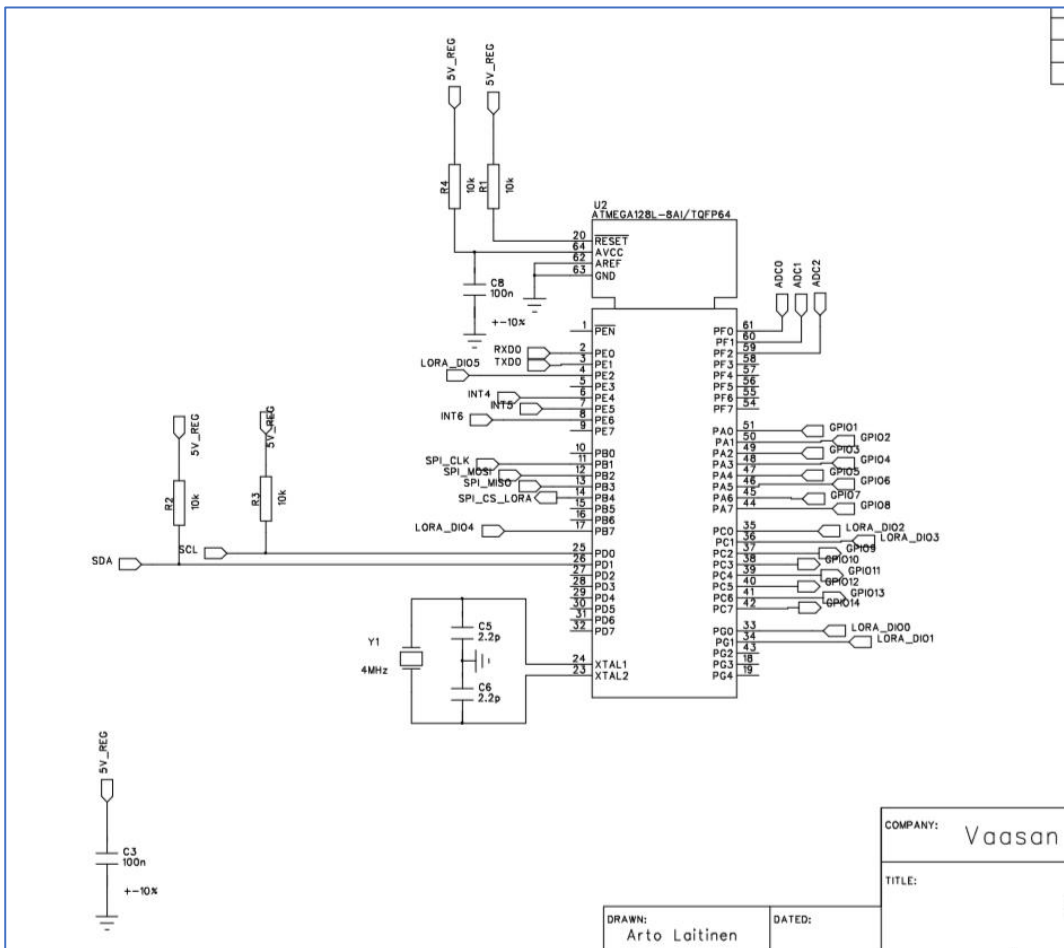


Fig.25 – Microcontroller pin diagram on the mainboard

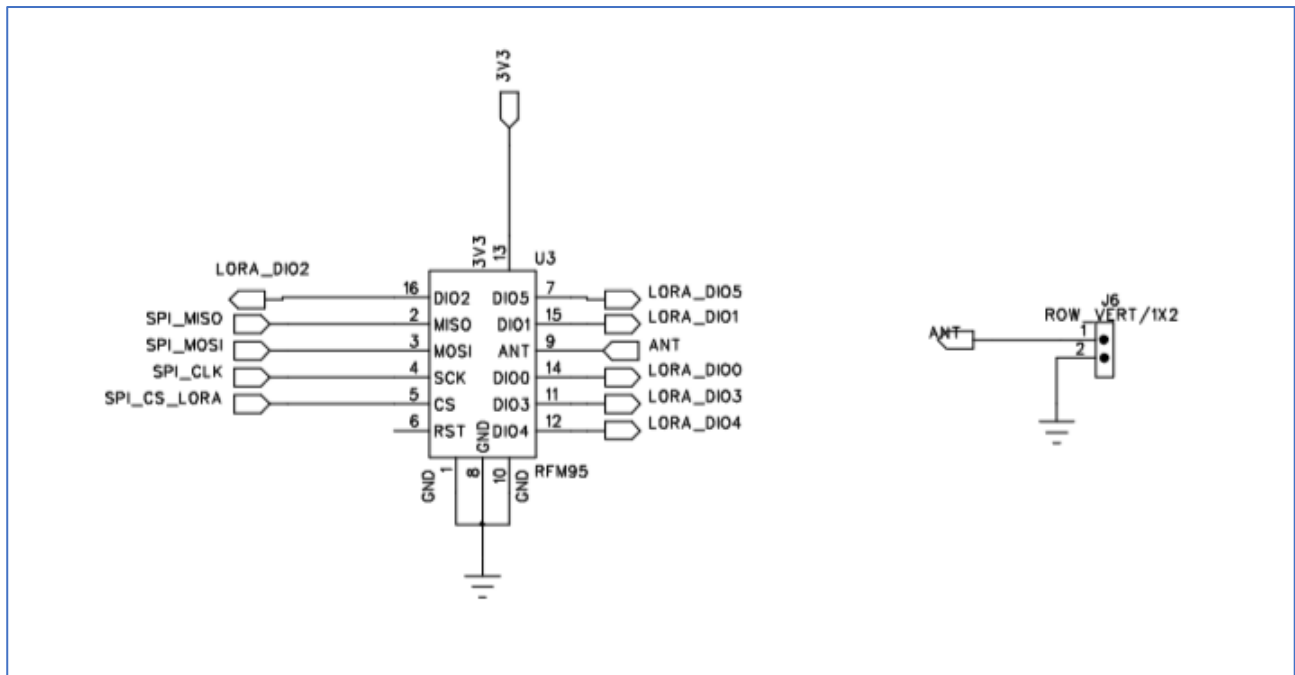


Fig.26 – LoRa Chip Module on the Mainboard and its pin diagram.

Component/Order List

The order list can be seen below, which includes the components and sensors ordered in order to build our node (see fig. 27).

Amount	Description	Code	Price in €	Order code	Shop
1	LIGHT-TO-DIGITAL CONVERTER	TSL2560	3.85 €	1226888	farnell
1	BAROMETER	MPL115A1T1	5.05 €	2674322	farnell
	light	TSL2561T	3.83	1226888	farnell
3	IR transmitter	OP293B	0.679	491299	farnell
3		GL480E00000F	3.45	9707794	farnell
3		VSLY3850	0.98 €	1870806	farnell
3		VSLB3948	0.64	2504157	farnell
3		TSHA4401	0.315	1045517	farnell
3		OP293A	0.9	1497874	farnell
	IR receiver				farnell
3		TSSP58038	0.61 €	2251445	farnell
3		TSOP58438	0.28 €	2251392	farnell
3		SD5600-001	5.21 €	1611518	farnell
3		HLC2701-001	3.39 €	1461621	farnell
5	MCU	ATMEGA128A	4.79 €	1773397	farnell
3	GPS	MICROSTAK GPS	21.47	2434228	farnell
5	LM2594MX-5.0	LM2594MX-5.0	3.61 €	9779841	farnell
5	MCP1825S-3302E/DB 3.3V LDO reg	MCP1825S-3302E/DB	0.49 €	1578404	farnell
20	68µF smd electrolytic capacitor		0.38 €	1850126	farnell
20	68µF smd electrolytic capacitor		0.35 €	2065959	farnell
5	100µH smd inductor shielded		2.69 €	1463475	farnell
20	1A Schottky diode		0.37 €	1843674	farnell
20	1µF MLCC X7R		0.31 €	1414042	farnell
20	22pF NP0 capacitor		0.04 €	1414678	farnell
5	Boost (Step Up) Switching Regulator	SP6641BEK-L-5-0	0.84 €	1317825	farnell
5	MPPT battery charger IC	LT3652EMSE#PBF	6.11 €	1839126	farnell
1	antenna for gps	W4000G197	11.17 €	1900077	farnell
1	antenna for gps	ADA-A720-S	43.90 €	1899494	farnell
1	antenna for gps	ANT-24G-S21-P5FL	10.82	2133445	farnell
2	a U.FL to SMA female adapter	R-132G7210100CB - RF	10.24	1699231	farnell
3	air quality	MIKROE-1630	14.41 €	2521695	farnell
1	air quality	MICS-5914	13.20 €	523-MICS-5914	mouser