UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

# Smart System Signalization Prototype for Flow Control of People in Crosswalks

**Chrysologo Rocha de Oliveira Neto**

Dissertation presented to the School of Technology and Management of Bragança to obtain the Master Degree in Industrial Engineering.

Work oriented by:

Professor PhD José Barbosa

Professor PhD Paulo Leitão

Professor PhD Luiz Fernando Caparroz Duarte

Bragança

2019-2020

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

# Smart System Signalization Prototype for Flow Control of People in Crosswalks

**Chrysologo Rocha de Oliveira Neto**

Dissertation presented to the School of Technology and Management of Bragança to obtain the Master Degree in Industrial Engineering.

Work oriented by:

Professor PhD José Barbosa

Professor PhD Paulo Leitão

Professor PhD Luiz Fernando Caparroz Duarte

Bragança

2019-2020

# Acknowledgements

Agradeço ao Instituto Politécnico de Bragança e a Universidade Tecnológica Federal do Paraná pela oportunidade de crescer como pessoa gozando do privilégio que é o programa de Dupla Diplomação. Agradeço ao Professor PhD José Barbosa por ter me dado calma e direcionamento no desenvolvimento desse projeto.

Agradeço a minha família e amigos por não terem me deixado desistir nesse grande desafio interno que foi estar aqui, por fim, finalizando esse documento. Em especial, gostaria de agradecer todo amor, amparo e conselhos dados por Bárbara Giehl, Dionizio Roman e Guilherme Gontijo. Vocês foram a base dessa minha vitória.

Gostaria, também, de agradecer aos meus queridos amigos Alessa, Goku, Hugo e Miko que, nesse período, resignificaram meu sentido de família e amizade. E, por fim, agradeço a todos artistas, poetas e músicos que interviram no meu ser e nutriram minha alma em momentos de dificuldade.

Gratidão.

# Abstract

Bad lighting conditions on crosswalks is a common problem on the urban environment. This scenery entail a large number of pedestrians fatalities and it shows a demand for solutions able to ensure their safety using lighting resources. The present work proposes a crosswalk's crossing process oriented by a system which is compound for a pair of Smart Devices that fit themselves in the Smart City idea. Inside this propose, they are capable to signalize for pedestrians the safe moment to enter on the crosswalk. These devices are a prototype of a system programmed in Python language and based in the Raspberry Pi and LoRa technologies. The work is split, mainly, on the hardware and software components development. In hardware level, it shows a circuit schematic design based on the Raspberry Pi Compute Module operating with the RFM95W LoRa module. In software level, it shows the incremental development of a Embedded System which reads inputs, gives lighting outputs and implements the communication, with encrypted messages, between the devices. Finally, this thesis shows a circuit schematic implementation wiled in KiCAD software and a embedded system focused in ensure well lighting and signalization on crosswalks. To validate the system are made hypothetical tests toward pedestrians behavior to cross the street on crosswalks.

**Keywords:** Smart Street Lighting; Crosswalk; Embedded System; Raspberry Pi; LoRa.

# Resumo

As más condições de iluminação em passadeiras são um problema recorrente no ambiente urbano. Esse cenário implica em um grande número de fatalidades e deixa evidente a demanda por uma solução capaz de garantir a segurança do pedestre usando recursos de iluminação. O presente trabalho propõe um processo de travessia em faixas de pedestres orientado por um sistema composto por um par de dipositivos inteligentes que se encaixam na ideia de cidades inteligentes. Dentro dessa proposta, eles são capazes de sinalizar para os pedestres o momento seguro para entrar na passadeira. Esses dispositivos são um protótipo de sistema programado em linguagem Python e baseado nas tecnologias Raspberry Pi e LoRa. O trabalho é dividido, principalmente, no desenvolvimento das componentes de hardware e software. A nível de hardware, ele mostra um projeto esquemático do circuito baseado no Raspberry Pi Compute Module que opera com o módulo LoRa RFM95W. A nível de software, ele mostra o desenvolvimento incremental de um sistema incorporado que lê entradas, fornece saídas de iluminação e implementa a comunicação, com mensagens criptografadas, entre os dispositivos. Finalmente, esta tese mostra a implementação do esquemático de um circuito usando o software KiCAD e um sistema embarcado focado em garantir iluminação e a sinalização nas passadeiras. Para validar o sistema são feitos testes hipotéticos em relação ao comportamento dos pedestres para atravessar a rua em faixas de pedestres.

**Palavras-chave:** Sistemas de Iluminação Inteligente, Passadeiras, Sistemas Embarcados, Raspberry Pi, LoRa.

# Contents

# List of Tables

# List of Figures

# Acronyms

**CC** Control Center. 17, 19, 25, 26

**CCTV** Closed-circuit Television. 25

**CM3** Compute Module 3+. xviii, 35, 46, 48–51, A2, B2

**EDA** Electronic Design Automation. 34

**ERSO** European Road Safety Observatory. 1

**H2H** Human-to-Human. 7

**IC** Integrated Circuits. 33

**ICT** Information and Communication Technologies. 11, 12, 16

**IDRA** Information Driven Architecture. 15

**IoT** Internet of Things. xvi, 5, 7–10, 12, 14–16, 19, 23, 25, 27, 28, 75, 76, 84–86

**LCU** Local Control Unit. 17–19, 76

**LED** Light-emitting diode. 17, 18, 20, 21, 58, 59, 64, 65, 67

**LPF** Low Pass Filter. 22

**LPWAN** Low Power Wide Area Network. 19, 27, 28

**LU** Lamp Unit. 16, 17, 19, 76

**M2M** Machine-to-Machine. 7

**NFC** Near-Field Communication. 19

**OOP** Object-Oriented Programming. 41

**P2P** Peer-to-Peer. 30, 36

**PCB** Printed Circuit Board. 29, 34, 86

**QoS** Quality of Service. 12–15

**RFID** Radio Frequency Identification. 15, 19

**RTC** Real Time Clock. 20

**SiL** Smart Indoor Lighting. 18, 19

**SL** Smart Lighting. 7, 19, 76

**SoL** Smart Outdoor Lighting. 18, 19

**SPI** Serial Peripheral Interface. 36, 48, 49, 58, 62

**SSL** Smart Street Lighting. 5, 7, 18–20, 23, 29, 76, 77, 85

**UWB** Ultra Wideband. 19

**VN** Virtual Network. 15

**WBAN** Wireless Body Area Network. 19

**Wi-Fi** Wireless Fidelity. 19, 22

**WLAN** Wireless Local Area Network. 19

**WPAN** Wireless Personal Area Network. 19

**WSN** Wireless Sensor Network. 15

# Chapter 1

# Introduction

## 1.1 Framework and Motivation

The pedestrians are the most vulnerable in traffic accidents during the daily process of locomotion. According to European Road Safety Observatory (ERSO), in general, 21% of all road fatalities in Europe are pedestrians [1].

When a comparison is made between all accidents that happens and those that occur with pedestrians, this vulnerability become visible. Around 5,5 of 25 thousand fatalities happens with pedestrians as shown in the Figure 1.1.

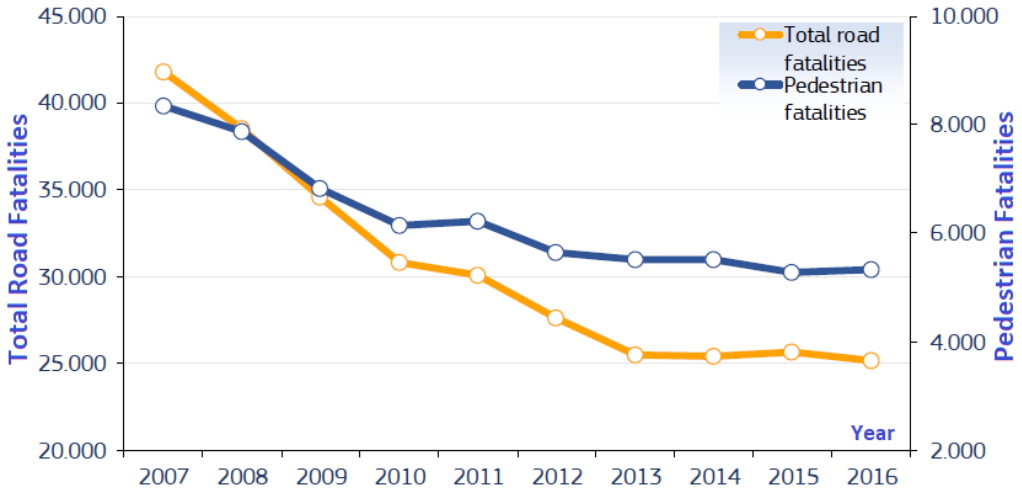



Figure 1.1: Number of pedestrian fatalities and all road fatalities, EU [1].

Bearing in mind when and who suffers, there is an alarming situation for fatalities with elder people and accidents happened in bad illuminated places. As shown in Figure 1.2, the majority of people in Europe dying in accidents have age between 80 and over 90 years old, with percentages of 43% and 45%, respectively.

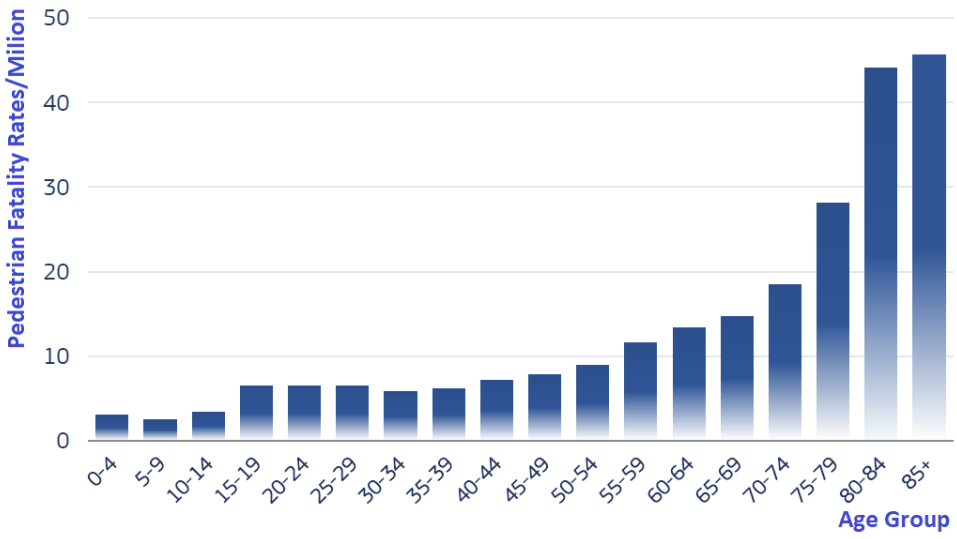



Figure 1.2: Pedestrian fatality rates per million population by age group, EU [1].

In Portugal it's even bigger, reaching almost 40% for who have over 64 years, as shown in Figure 1.3.

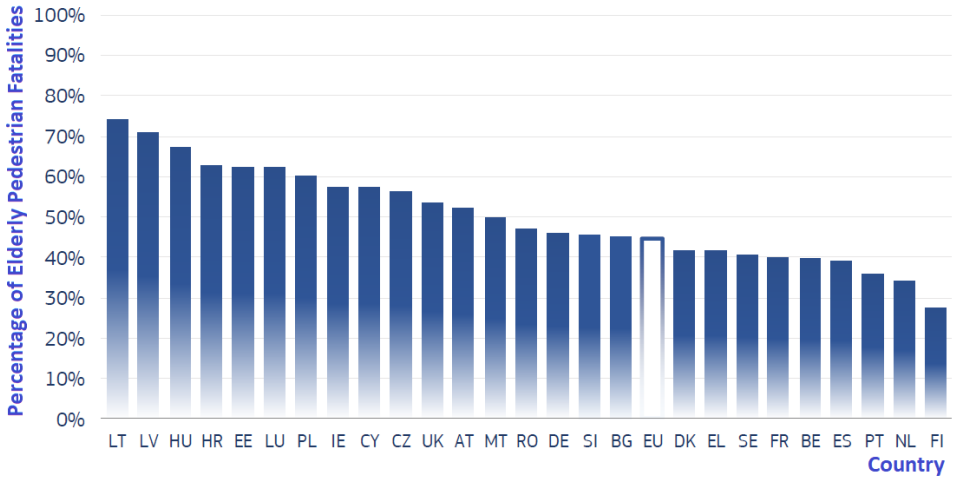



Figure 1.3: Percentage of elderly pedestrian fatalities (age>64) of all pedestrian fatalities by country [1].

In a Portuguese overview, as show in Figure 1.4, almost 40% of all pedestrian deaths are in bad lighting conditions.

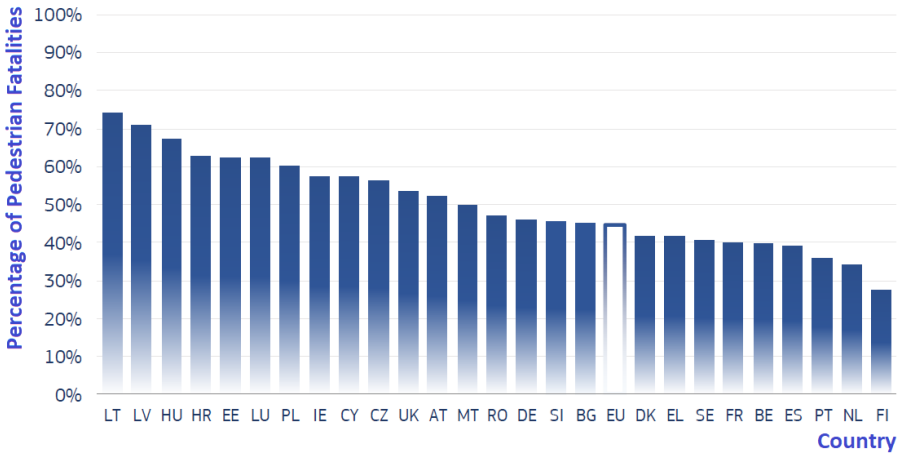



Figure 1.4: Percentage of pedestrian fatalities during darkness of all pedestrian fatalities, EU [1].

Finally, in addition of all these numbers, Figure 1.5 shows that the larger accident causation, resultant in death, happens because almost 40% of pedestrians, not the drivers, have a premature action in the roads.

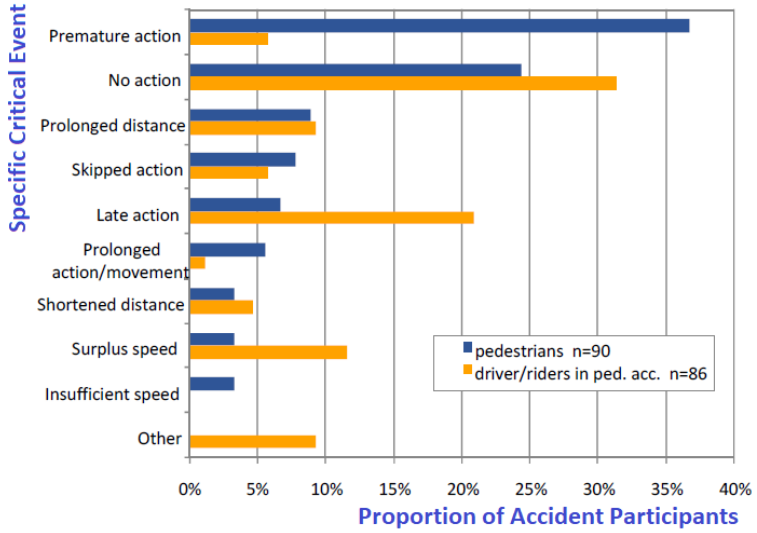



Figure 1.5: Distribution of specific critical events - pedestrians and driver/riders in pedestrian accidents.

All these estimates indicate that, in Portugal, a delicate situation is happening in traffic accident scenery. The Figure 1.6, shows that the elderly population is growing and currently represents 21,8% of total Portuguese population and, until 2080, will grow even more [2].

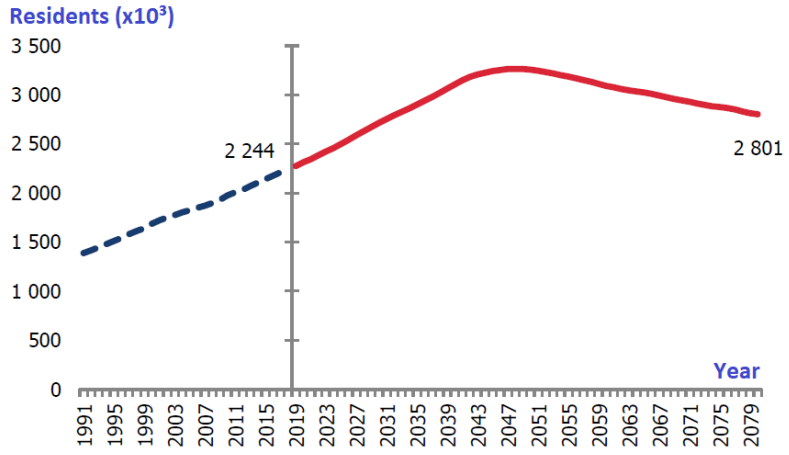



Figure 1.6: Elderly resident population (age>65) in Portugal from the years 1991 to 2019, with a projection until 2080 [2].

They represents a important portion of this and they need a special attention. Independent of the age, in general, it's necessary develop smart tools to prevent premature actions and give accessibility in pedestrian traffic. It is important to structure a way to make safe the people flow in bad lighting places, mainly in the most desprotected moment of a pedestrian, the street crossing.

Thus, this work addresses the creation of a smart lighting system applied in crosswalk signalization. It includes a prototype with the circuit schematic design and an embedded system proposal.

## 1.2 Objectives

The main objective of this work is to develop a embedded system able do control the people flow on crosswalks ensuring lighting and secure. In this way, the other objectives also make up this work:

- Propose a Smart Street Lighting (SSL) model applied in crosswalks able to ensure lighting and security for pedestrians.
- Develop a smart device system that is Raspberry Pi and LoRa based.
- Develop a embedded system based on Python language.
- Implement a schematic circuit which the embedded system can be incorporated.

## 1.3 Document Structure

This document is divided into 6 chapter which they show where this thesis proposal are inserted, how it has been developed and where the final results can be applied.

The work introduction is discussed in Chapter 1 and it shows the work's framework and from where comes the project motivation. This chapter shows statistics which sustain the hypothesis that pedestrians suffer in traffic accidents with bad light conditions. Based on that, it shows a rising demand for a system able to ensure secure and lighting on crosswalks.

Chapter 2 presents basic theoretical ideas about IoT and Smart City and how a Smart Device can be integer on them. In sequence, describes the Smart Lighting working structure and how it is framed on Crosswalks, explaining about Smart Street Lighting and its components. In the end, it is explored 7 work papers which give architectural references and relevant topics to choose an approach and make the Smart Device implementation.

Chapter 3 contains a system proposal's methodology inside the crosswalk's lighting problematic context. It structures the idea behind the project and how this project works a Hardware and Software level. In the Hardware level is described how the inputs and outputs should work with the control board, besides what compound each hardware peripheral. In the Software level is explained how the algorithm for the Embedded System works and what tools are used to develop it.

Chapter 4 explain the hardware schematic and split it in all components which compound this work. It also presents all the steps in the software development, the created

versions and how they are integrated to implement the Final Embedded System. In the end, it explains how incorporate this software in a hardware Raspberry Pi based.

Chapter 5 explores the Final Embedded System, test its work with ordinary/extraordinary crossing cases and discuss scenarios where the schematic can be helpful. Finally, the Chapter 6 exploit the results, makes resolutions about each chapter and proposes future works with this Smart Device implementation.

# Chapter 2

# Related Works

In this section will be analyzed related works in the area of Smart Lighting (SL), first, describing relevant topics about Internet of Things and Smart City, constructing these themes basic ideas. After this, will be specified SL branch called SSL, describing their features and real implementations emphasizing relevant tools applicable on crosswalks.

Bearing in mind the application developed during this work and encompassing into the current context where Smart Lighting is an important strand in Smart City's environment, this chapter tries to construct, based on a scientific research, the state of the art.

## 2.1 IoT Environment for Smart City

### 2.1.1 Internet of Things

Internet of Things has been gained ground in the technology projections for the second decade of the 21st century. It started in 90s and, ever since, changed the concept of what is an object and how it can behave. The IoT paradigm expand the internet usual capabilities from Human-to-Human (H2H) for the Machine-to-Machine (M2M) communication allowing the creation of smart and connected products and, in another words, it provides a connection among everything in anyplace [3].

The IoT, by default, have a workflow with an object sensing of specific information,

a triggering of an action and a feedback to the administrator about the current status of the requested attitude.

Figure 2.1 shows the process and it starts from the object sensed about, for example, chemical changes in the air. This modification generate data that could be understand like a specific object information.

The combination of different sensor types structure the design of smart services. In second step, a smart device starts to operate performing an action gamma triggered by the information received and, finally, this system provide a feedback with taken decisions monitoring the details.

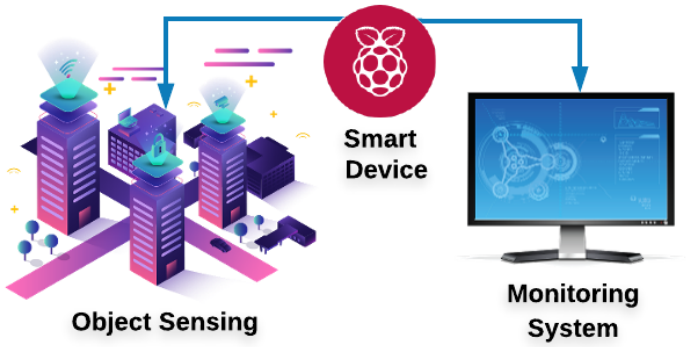



Figure 2.1: Basic workflow used in IoT [4].

The majority of IoT systems have a generic architecture that provides the aforementioned operation. Therefore, for describe grossly this architecture, it is necessary a layer analogy, because in each one, a different type of operation happens. According to Tan and Wang [4] this structure have layers relation as can be seen in Figure 2.2.

That five layers can be briefly described as follows:

1. Perception: It is compound by the detectable physical object and a sensor device able to make specific information acquisition. All type of sensors can be used to capture interest environment data.

2. Network: Basically, the function of this layer is transfer information from Perception to Middleware layer safely. There are many different ways to make a communication

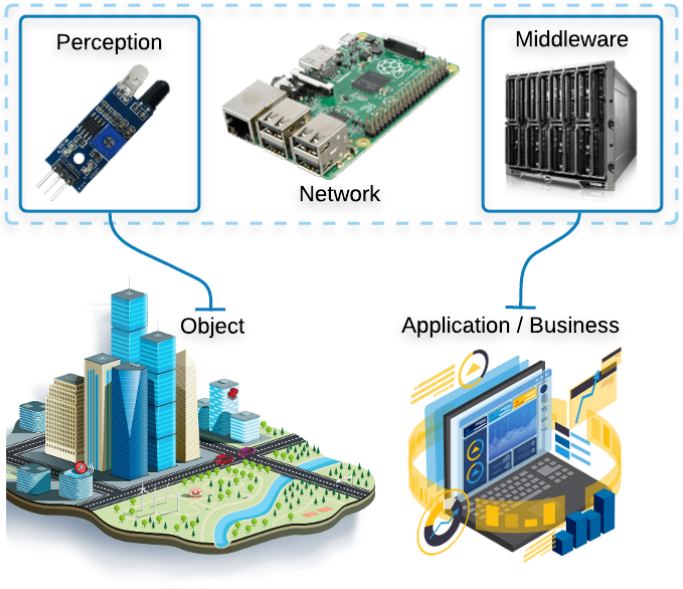



Figure 2.2: Generic Architecture for IoT System [4].

channel, wired or wireless, some of those can be 3G, Wifi, Bluetooth and LoRa, depending on the purpose.

3. Middleware: This layer is responsible for service management and has link to the database. Beside that, it get the information from the network and make them storage. Each device has its own operating characteristics and can be, for example, a microcontroller.

4. Application: Based in the objects information interpreted for the Middleware layer, this layer provide a system complete overview.

5. Business: Seen as a layer that determines the smart solution success, it builds business models, statistics and flow charts, refining information quality and even giving behavior predictions of detected objects.

Thus, to create smart devices able to provide these services are necessary structural requirements such as automation, intelligence, dynamicity and zero configurations [5] as described below:

1. Automation: Objects should be able to handle themselves automatically. Collecting contextual data, processing it, collaborating with other required objects and acting according to the condition should be at some level of automation.

2. Intelligence: Devices should be able to perform intelligently. Intelligence in devices give them power to act based on different situations.

3. Dynamicity: Should be available to achieve the IoT concept. When one devices move from one place to another or one application domain to another, it should be dynamically recognized in new scenarios.

4. Zero configurations: Many users are not expert who uses the physical devices. So, users should be able to configure devices and develop services of devices without having much technical details.

Thus, IoT smart devices need to perform a lot of tasks inside Perception and Network layer to achieve these four goals. They are commonly developed with a lot of electronic modules, e.g., wireless communication circuits, actuators, sensors, power supply control and microcontroller/microprocessor.

All of them should work in group and need to be controlled by the only one "brain" component, the microprocessor. It gives commands to another modules based in a firmware which operates logically using programming languages such as C-language or Python [6].

All these features has been reinventing the current internet concept into an ubiquitous network where interconnected objects, besides harvest information from the physic environment and react on it, can use this traditional internet for information transfer, analytic and applications [7].

### 2.1.2 Smart City

In 2014 almost three quarters (72,5%) inhabitants of Europe Union were living in urban places like cities, towns or suburbs [8]. In global context, estimates demonstrate the ratio of the world‘s urban population grown up 55% in 2018 (4.2 billion people) and can achieve

68% by 2050 [9], thus, that population will nearly double. For organize this huge people concentration, the Smart City concept propose an improvement of two ideas: 'Smart' and 'City' [10].

Within networked Information and Communication Technologies (ICT) idea, there is not a consensus about the Smart concept, however, it is possible to get an approximation defining like almost anything considered to be modern and intelligent. But, applying in a system, there is a concise vision of smartness where a servant is surrounded by another servants, people or devices, which assemble an ecosystem, embedded within "servant systems" [11]. Another key concept is that cities are considered a range definition for people agglomeration inside an urban area.

Connecting both concepts, Smart City is framed as an urban system that aim smart solution around infrastructure, economy, governance, environment, services, mobility, living and people, as shown in Figure 2.3. The development inside that eight components helps to measure city smartness, therefore, all these services have been improved using telecommunications technologies.

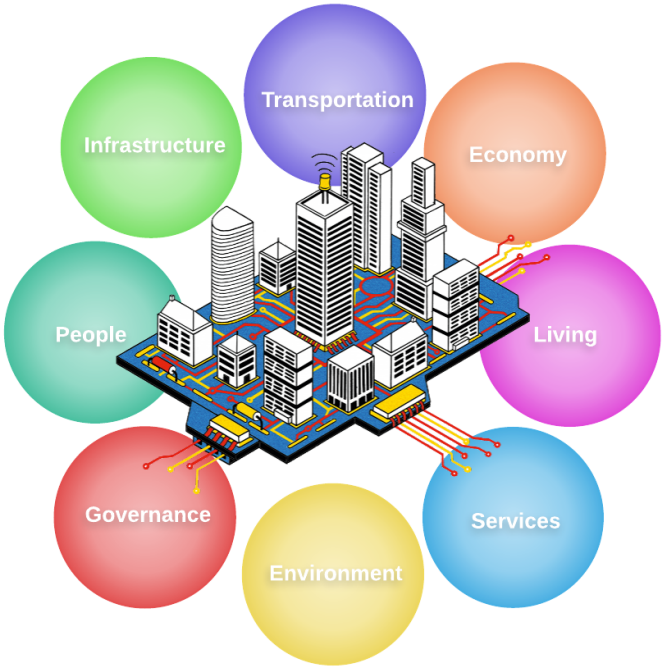



Figure 2.3: Smart City eight components [11].

"Infrastructure", like water and energy networks, can be monitored by sensor or smart grids. "Transportation" can has real time localization feedback of public transport vehicles. "Environment" can receive help inside natural resource protection and management using, for example, drones and data sent by image processing smart system.

Applications inside health, education, tourism and safety provide Smart "Services" enhancing Living life quality. People engagement expand with ICT good use, thereby, help "Governance" to be more pragmatic and effective. All this structure strengthen the "Economy" by business development and "People" can "Live" with wellness, upgrading they creativity and opening space for innovation [11].

These innovation possibilities, based eight pillars, have been amplified with IoT paradigm growth. Inside that context, this technology are expanding increasingly toward a fully integrated future internet with huge functionalities variety to improve urban population wellness.

### 2.1.3 Architectures

In this Subsection will be described network architecture concepts for the new internet, exploring four most common architectures aimed to Smart City construction. They can be autonomous, ubiquitous, application-layered and service-oriented, which will be explained their individual characteristics, practical implementations examples and Quality of Service (QoS) requirements [7].

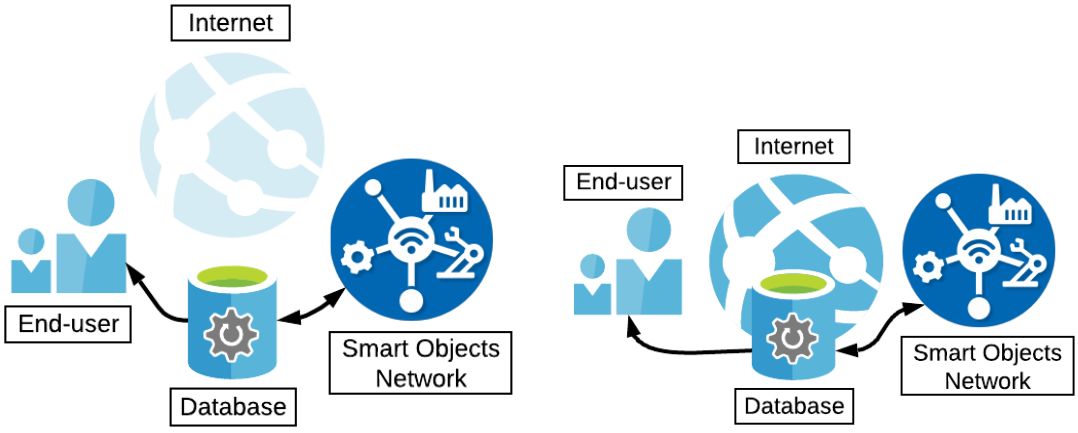
All these IoT networks types have the same issue to be solved, a lot of different smart objects networks applied at to many specific implementations. In most cases they are structured using different protocols, thus, it demands a powerful gateway able to translate all these heterogeneous data and put it, neatly, in internet connected data-base. So as follows will be gives their brief description, practical implementations and QoS issues and goals.

**Autonomous**

- Description: As the name define, it is not connected to public networks, and is very common in reality, as shown in Figure 2.4a. In contrast, this connectivity is not prohibited and, usually, a gateway can be used. In general, IP protocols are used, but it is not mandatory, however, helps to its scalability and flexibility, furthermore, huge address spaces are possible to be employed.

- Practical Implementation: Parking Grounds, several times, have an autonomous network. For example, it is desired to give for final user how many parking spaces are vacant or not on desktop computer. It is possible using a sensing structure for each parking space and all of them send this status for network server, responsible to application database updating and give information to final user.

- QoS: In quoted example, sensor precision and system responsiveness are head quality goals. However, this is completely function-dependent, because each application has an unique infrastructure, therefore, each one have their own quality factor determinant.




(a) Autonomous Network. (b) Ubiquitous Network.

Figure 2.4: Diagram comparison between (2.4a) Autonomous and (2.4b) Ubiquitous network architecture.

**Ubiquitous**

- Description: Pretty similar to Autonomous Network, however, there is a punctual differences between them, the server structure, as outlined in Figure 2.4b. Final user is able to access network updates from an internet connected database, furthermore, with real time updating.

- Practical Implementation: Bearing in mind smart lighting home projects, it is easy to construct the idea. Using home network and smart devices able to detect voltage, current and to map the light system. It is possible to create statistics for energy use control, automated lighting programs and user remote use by smartphones. There is a lot of possibilities, but the point is how easy different users can enjoy of it by different places.

- QoS: While multi-access is a key advantage, it is also the biggest misfortune. Within the described scenery, the standard end-to-end protocols can have a good behavior with the huge diversity of QoS requirements. Besides all advantages, this complex network dynamic puts this approach in the face of a great deployment challenge.

**Application-Layer Overlay**

- Description: It is very usually use data acquisition from thousand of independent nodes in IoT systems. In other wise it is a several problem for QoS. According to [12] "An overlay network is a logical network built on top of one or more physical network".

  To implement it trying to solve these QoS problems, this approach uses concepts of Network Virtualization. When formed an application-layer overlay network, the process starts to happens based in selected nodes performing in-network data processing tasks.

- Practical Implementation: This structure is commonly used for Smart City Environment monitoring as a tool to sense micro climate variations, air purity and their

chemical components.

- QoS: Scalability, Isolation and Programmability are just some features provided by it. Coexistence of multiple networks, including isolation of each Virtual Network (VN), it is based in their programmability, customizing protocols and deploying diverse services. In contrast, their physical structure, lot of times, can destabilize the virtualization process.

**Service Oriented**

- Description: Service Oriented Networks purpose the use of Information Driven Architecture (IDRA) to seek IoT quality goals using an approach focused on network service, which is able to perform different network demands, such as addressing, routing and synchronization applied to Wireless Sensor Network (WSN). A more detailed idea can be consulted in [13].

- Practical Implementation: It includes topics such as disaster intervention, health monitoring and industrial process monitoring. Imagining an hospital environment, there are some variable, such as blood pressure, heart-beating and biochemical levels, that can be monitored by smart medical objects using an wireless connection between them.

  While all this data are detected, a Radio Frequency Identification (RFID) can controls people flow of specific patients rooms. Even more, the lighting and temperature can be controlled remotely by nurses or, automatically from a previous determined routine. Lastly, only one gateway manage these features. This is the challenge which IDRA proposes to overcome.

- QoS: The challenges have a direct ratio with QoS. If the goal is unify the services on the same platform, it starts to happening a network congestion. Therefore, it implies several packet drop which reduces the network performance.

## 2.2 Smart Lighting

After understand IoT idea through Smart City environment it is important to contextualize where Smart Lighting are framed. According to [14] "Smart Lighting comprises an heterogeneous and multidisciplinary area within illumination management, with the possibility of integrating a wide set of sensor and control technologies, together with ICT".

### 2.2.1 Working Structure

An overview about this system type is structured by an intelligent sensor-equipped lighting able to communicate itself with other and a management center. Seeking to reduce power consumption, maintenance costs and natural environment preservation, these systems use an architecture branched in three pillars. This lighting smart device integration can be seen in Figure 2.5.

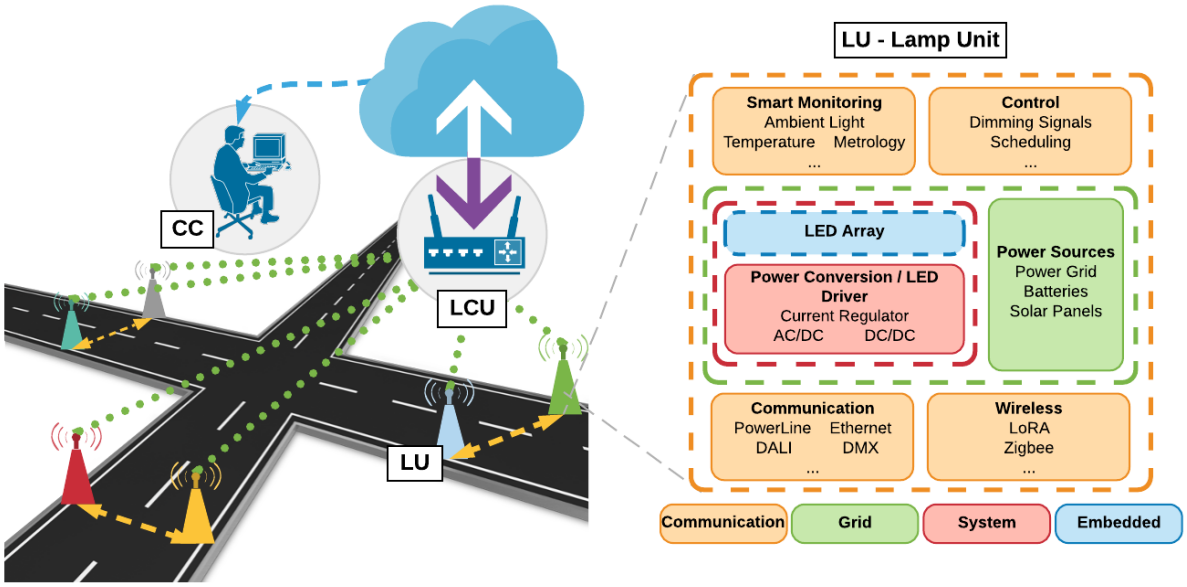



Figure 2.5: Overview of SL and their LU integration levels, adapted from [15] and [14].

Its overview [15] can be described concisely as follows below:

- Lamp Unit (LU): Beyond the illumination feature, this has a controller endowed

with a lot of sensors. They are responsible for data acquisition and are able to communicate (wired or wireless) with another LUs which, in general, send information to a gateway. Recently have been happening lamps manufacturing process improvements and it started to use Light-emitting diode (LED) in lighting, reducing $CO_2$ emissions and energy consumption [16].

- Local Control Unit (LCU): This control unit collects data from LUs and redirect to a server using, as much to collect as to redirect, long (LoRaWAN, Sigfox etc.) or short range (ZigBee, 6LoWPAN or Bluetooth Low Energy etc.) communication protocols. This choice depends on the environment which they are implanted, respectively, whether it is remote or an easy access place. In many cases it can interface a mixed network topology and must ensure a good communication between the LUs.

- Control Center (CC): Control Center receives data from LCU and store it in a server, usually, using cloud services. In general, urban areas have a huge data volume acquisition from various LUs. Furthermore, the CC can uses data analysis tools to implement smarter decisions and improve administration process.

And these LU integration levels [14] as:

- Embedded: Lighting structure or properly the light source.

- System: Responsible for voltage/current adjustments and energy conversions.

- Grid: Monitoring of power sources and energy generation/distribution schemes.

- Communication and Sensing: This level contain the complete lighting solution encompassing sensors, signal acquisition, communication platforms, supplies and embedded system.

The architecture, shown in Figure 2.5, can be adopted as centralized or distributed. The difference between these approaches is when communication between LU and LCU fail. In the first case, the data sent can be lost because all the system is controlled from

one location. In the other case, it is redirected to an available LCU because the system structure is modular [15].

Another framework that can be seen in developed works are Indoor and Outdoor lighting. The Smart Indoor Lighting (SiL) is applied in public service buildings, big factories and companies (few times using wired technologies). It tries to save energy controlling lighting duration, and adjusting light intensity near windows for natural lighting usage optimization.

The Smart Outdoor Lighting (SoL), which is this work's focus, usually use wireless deployment systems. It tries to be smart enough to work without human intervention, and it seeks energy efficiency based in weather conditions, availability and sustainability. It also uses presence sensing, images acquisition and climate variables [17].

### 2.2.2 Smart Street Lighting

Thus, deepening outdoor scenery and such as described in Chapter 1, it is known that electricity has a huge demand on cities and smart infrastructures. However, they offer smart paths to achieve street lighting improvements, such as energy efficiency, interactive safety zones, context awareness, privacy and security and system expendability [18].

Power consumption in SSL consist in lighting resource used at the correct moment with correct type of lamp, currently, LED based bulbs are usually employed for it. A good example of energy safe are lighting systems helped by a sensing interface, which are able to control luminosity and know the exact time that someone want to use the crosswalk.

Still thinking in the before mentioned example, SSL tries to supply user demands with regard to well illuminated zones. In this case, the crosswalks lighting zones can be expanded or reduced according the implementation. Sometimes, this size variation can works with interactive possibilities, over all, trying to ensure safety.

Context awareness is a feature that structure the idea of a continuous back-ground

operation. The system is available during all time for end-user demands and, if necessary, continue to operate in a request independent mode. It works sending error reports, generating estimates or responding to automatic functions, e.g. lighting intensity.

In the end, it is supported by a cloud based server which seek IoT criteria and try to guarantee data anonymity and block unwanted system manipulation. The SSL pursue systems deployment that has quick and easy integration in future approaches using software components able to fit in constant updating of Smart City needs.

### 2.2.3 Remote Communication

In Smart City context are used short and long range technologies to implement smart solutions and this include SL. Bearing in mind short range technologies, the work [19] proposed promising technologies which have been explored from their year publication to the present days works [20], such as Bluetooh, Wireless Fidelity (Wi-Fi), Zigbee [21], HomeRF, Ultra Wideband (UWB), RFID and Near-Field Communication (NFC).

These technologies are applied in Wireless Local Area Network (WLAN), Wireless Personal Area Network (WPAN) and Wireless Body Area Network (WBAN). They are useful in personal communications and SiL projects, few times already was used to SoL. However, throughout the years, have been losing space to long range technologies, because considering outdoor spaces, greater distances are desirable. Therefore, this technologies are most common to use for communication between LCUs and LUs in SiL, due distance that reaches less than 100 meters [15] and, in general, less than 1000 meters [20].

In another side, IoT communication also requires long range tools which make feasible, for example, a large urban area coverage that needs a lot of LCUs sharing information between them and with a central CC from some kilometers of distance. This type of necessity fits in Low Power Wide Area Network (LPWAN) that, according [20], [22] and [23], has been mainly used technologies such as LoRa [24], Wi-SUN [25], SIGFOX [26], RPMA [27], Weightless [28] and DASH-7 [29], and they are also applied to IoT-Enabled SSL.

Lastly, all of them have their own space satisfying goals, such as security, coverage, performance in non-line of sight conditions, network topology, business model, implementation/deployment/operation complexity, data rate for up and downlink costs and other aspects [22].

After this overview in these essential themes around Smart City, Internet of Things and Smart Lighting, the next section will present seven works published since 2013 until the current year (2020).

## 2.3 Developed Works

These papers present references sources that have given direction to a good work considering energy efficiency, communication method and pedestrian behavior. The next subsections, named according each works' title, will describe adopted system architectures for each one and their results focusing in the interest points used to develop the Chapter 3.

### 2.3.1 Smart Street Lights

This paper implement a SSL to save energy and to lighting streets at night. To achieve it, this work proposed a LED street lighting and a road movement detection. The lighting is activated each time the ambient light intensity becomes under a light intensity threshold and happens a laser detection (during night period) based in a Real Time Clock (RTC) [30].

#### Architecture

This system architecture have been planed as block diagram shown in Figure 2.6 which is mainly compound by a sensor block, control block and lighting block.

Sensor block is integrated by an Ambient Light intensity sensor and an array of Laser

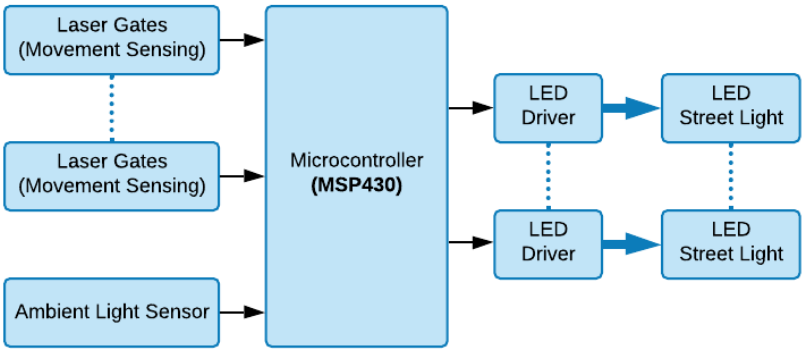



Figure 2.6: System architecture block diagram, adapted from [30].

Gates do detect movement on the road. Control block use MSP-EXP430F5529 experimenter board [31] to control sensor inputs and send signals to LED driver according ambient variations or car detection on the road. Lastly, Lighting Block is compound by an array of LED and an LED drive which use a Buck Chopper with 2 fixed PWM signals generated using two timer modules in NE556 for light intensity control.

#### Relevant Topics

Light intensity control based in ambient luminosity using microcontroller and electronic circuits are an approach to save energy. This work [30], directly propose an useful LED driver and ambient light intensity circuits.

### 2.3.2 Development of Cloud Based Light Intensity Monitoring System Using Raspberry Pi

This work introduces real time remote light intensity monitoring system of a indoor environment using Raspberry Pi, which the information are stored in a cloud database [32].

#### Architecture

System architecture have been planed as block diagram, shown in Figure 2.7, and it is mainly compound by a signal conditioning circuit and a raspberry pi responsible to data

storage in a cloud. This data is used by a system which, in this case, gives light intensity (LUX) in a report based in some acquisitions during a specified day.

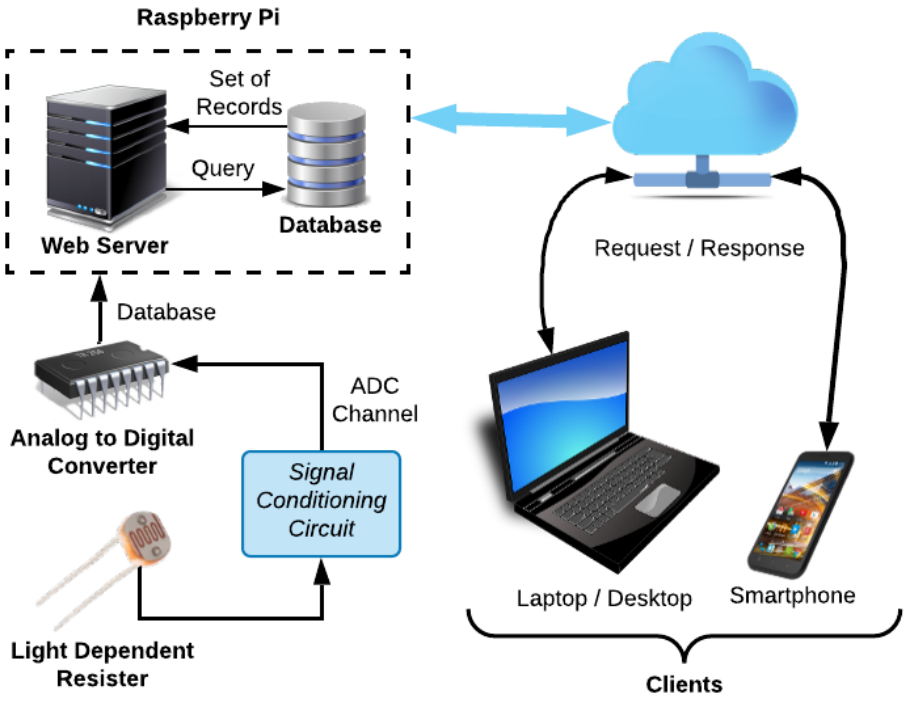



Figure 2.7: System architecture block diagram, adapted from [32].

### Relevant Topics

This paper also proposes light intensity detection circuit using a different approach when compared with [30]. This circuit use a 3rd order analog Low Pass Filter (LPF) which is used to reduce unwanted high frequency signals from flickering lights, glare, pulsating light sources. Furthermore, it proposes a SQL storage in Raspberry Pi that use Wi-Fi communication to put in cloud all data acquired so, in this path, gives good references to structure an Webserver idea.

### 2.3.3 A real-time sensing system of elderly's crossing behavior at outdoor environments

This study [33] gives a real-time sensing system that is mobile based destined to elderly people. So, different of [30] and [32], its relevant topics is not the architecture, but rather it specific study in elderly pedestrians' behavior, which provides references to understood and implement validation tests for one of motivations described in Chapter 1. Another relevant point is a street-crossing behavior standardization, that can be seen in Figure 2.8, which also enriches better services chances in a smart system.

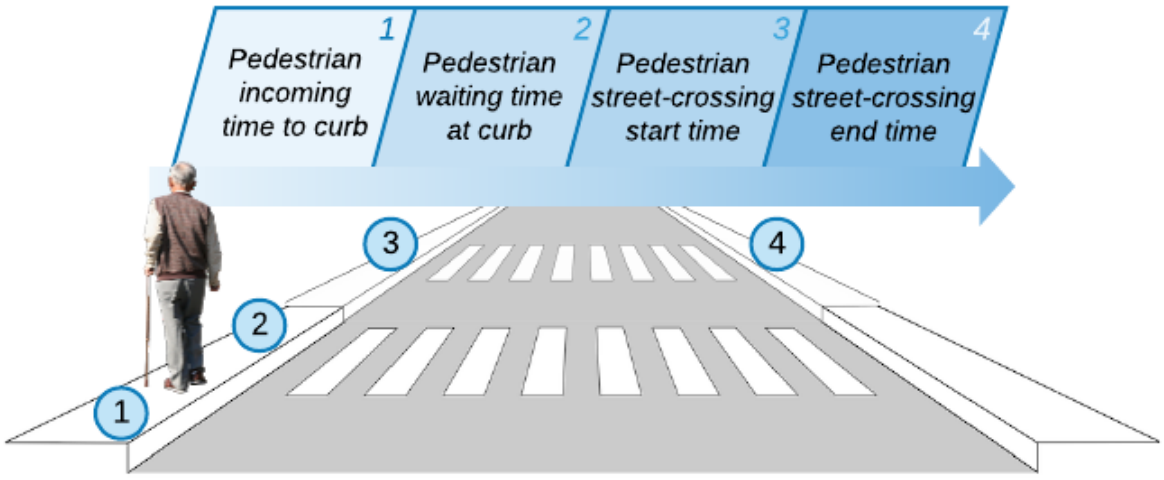



Figure 2.8: Pedestrians' street-crossing behaviour process, adapted from [33].

### 2.3.4 Automation Control and Monitoring of Public Street Lighting System based on Internet of Things

This work [34] propose a system SSL focused in low maintenance and accuracy using IoT concepts. It uses Ubidot [35] as a cloud IoT Application Enablement ecosystem and Raspberry Pi, to make hardware control of four enabled lamps. Thus, reads energy used, lamps condition and connection condition giving how much delay the microcontroller needs to send real light status.

### Architecture

System architecture have been planed as block diagram shown in Figure 2.9, which is mainly compound by a device zone and it uses a microcontroller system based on Raspberry Pi. This system has a power supply, sensor systems, energy meter, telecommunication system and lamps connected to an output module.

Still in hardware, there is a control zone which use a laptop that access a GUI system based on Ubidots application. The software is based in Python language program that runs continuously and makes a system able to share data results of energy used, lamps condition and connection condition.

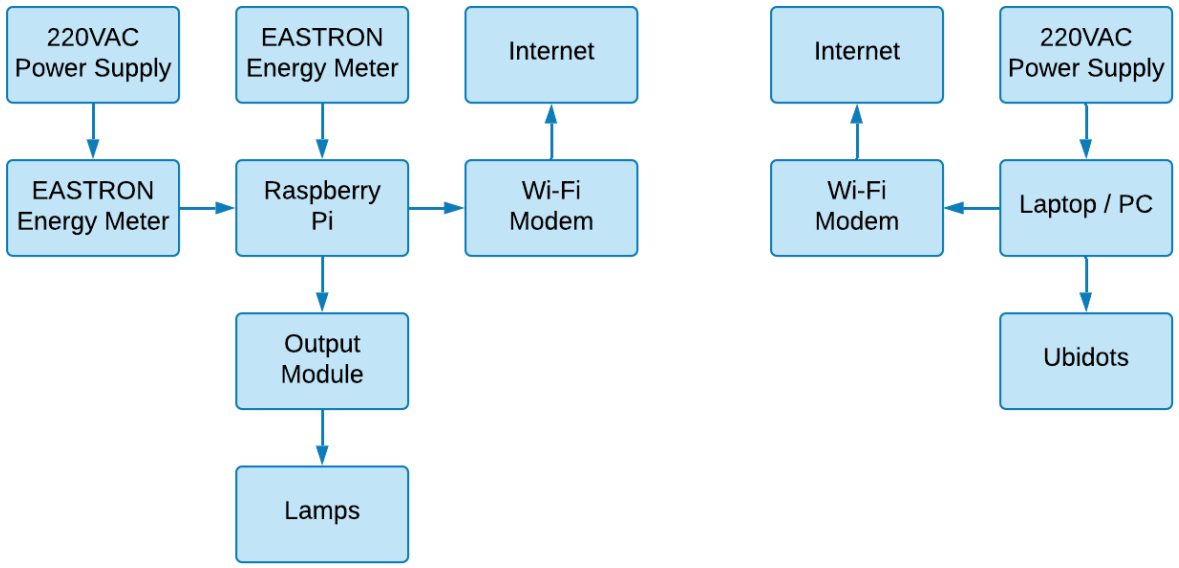



Figure 2.9: System architecture block diagram, adapted from [34].

### Relevant Topics

Easy maintenance methods are a valuable tools for a system that can has a huge amount of nodes scattered in a city. Thus, this project describe the use of an energy meter to identify a lamp correct working using difference between energy values before the lamp is activated and after it is turned off.

This tool can be employed to another devices for working status, thereby, make possible a system endowed by modules with well working reports. Furthermore, it uses an IoT server platform which can gives direction of how to make linkage between the obtained data by a hardware and a cloud server.

### 2.3.5 Pedestrian-Safe Smart Crossing System Based on IoT with Object Tracking

This paper [36] describe a system to prevent and detect accidents endowed with an rescue tool for an injured pedestrian. In general, it uses a floodlight to make the pedestrian, crosswalk area and obstacles more visible to driver and also use a a sensor group, including cameras, to vehicles and pedestrians tracking. In accidents situations, it uses a communication module to send a signal and a recorded video to a CC which can take appropriate measures to help the injured person.

#### Architecture

System architecture have been planed as block diagram shown in left-top image in Figure 2.10. It is mainly compound by seven components which are sensor unit, Closed-circuit Television (CCTV) unit, traffic control unit, network unit, control unit and control center.

Looking the right-top image in Figure 2.10 can be seen a scale down prototype implementation. And the other two bottom parts describe, in the left, a detection and tracking method for pedestrians and objects, while, in the right, shows an algorithm to tracking on the crosswalk.

#### Relevant Topics

This project makes several components good match in order to implement a safe crosswalk system. Sensor unit, divided in vehicle and pedestrian detection, using a range finding, laser-receiver and an ultrasonic sensor matched with CCTV, become possible a lot of detectable situations. Furthermore, this unit use a heart-bit signal to CC in order to

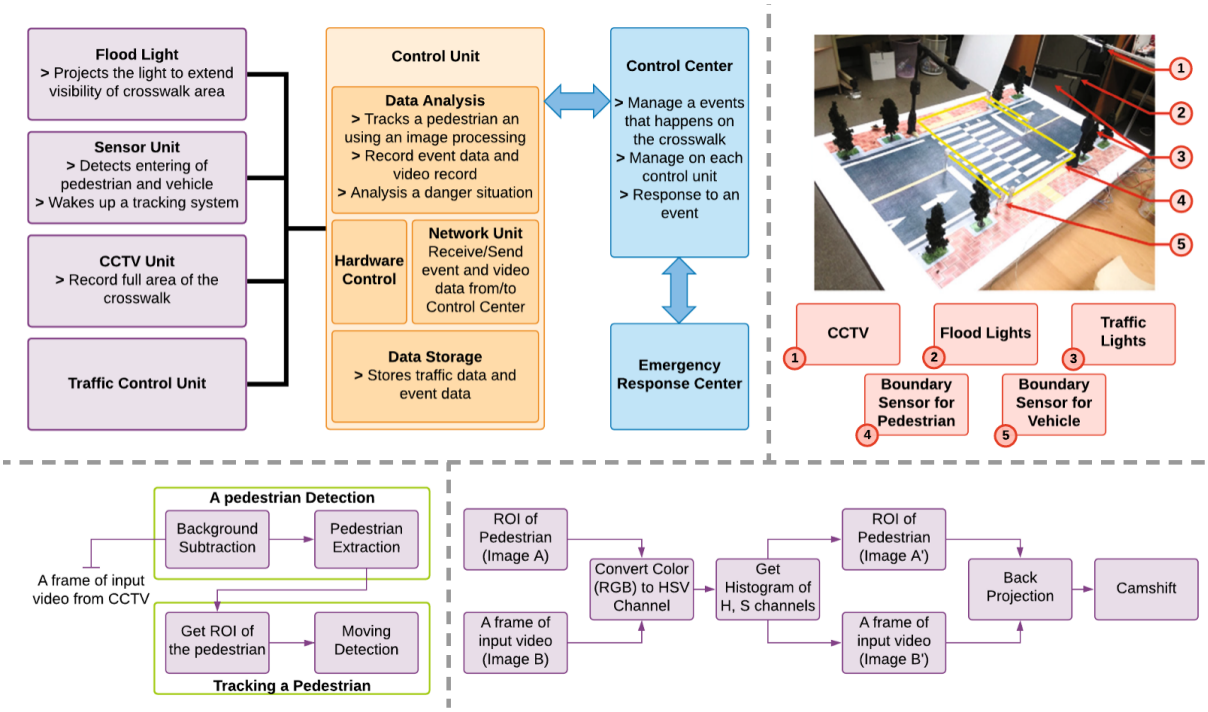



Figure 2.10: System architecture block diagram, adapted from [36].

detect activation failures enforcing reliability.

Traffic control unit propose, beyond floodlight and light controller, an extra signal to alert drivers in accident circumstances. Network unit is used to notify a CC about general information in crosswalks, like tracking and detection, processed by control unit algorithms. In the end, all system has key points which can be used as test reference.

### 2.3.6 Smart System of a Real-Time Pedestrian Detection for Smart City

This research [37] makes a Raspberry Pi microcomputer deployment as end-device directed to recognizing of pedestrians in real time and introduce additional functions to traffic lights, such as pedestrian's posture and movement detecting around a specified area.

#### Relevant Topics

In this work the interesting subjects are not focused in an architecture like another papers previously described, but in how computer vision tools can enrich solutions like, for example, the system that has been seen in Subsection 2.3.5.

Using pattern recognition helped by deep learning, big data and cloud services there is a lot of systems created to identify humans and this work cite and gives references of interesting works and tools [38] [39] [40] to makes this identification real.

So, the focusing points are recognition of pedestrian's intending to cross the road, the certain zone where they can be and their postures for additional verification. To achieve this goals, it tests a real implementation using Raspberry Pi 3 with Linux installed, Python 3 and the OpenCV, NumPy libraries, and the open-source TensorFlow library.

Thus, looking this deepening in image processing to identify pedestrian's behaviour it can be seen like a motivation to use Raspberry Pi to develop end-devices and how it can be a good platform to use complex algorithms, mainly when the goal is make improvements in transport and pedestrian traffic with security and automation.

### 2.3.7 Long-Range Communications in Unlicensed Bands: The Rising Stars in the IoT and Smart City Scenarios

This paper [23] makes an overview about LPWAN and how LoRa is a good representative choice based in theoretical view and using experimental deployments which implement a coverage range analysis.

#### Architecture

LoRa private network has been implemented using the architecture described in the block diagram shown in Figure 2.11 and has been deployed in Northern Italy. This LoRa coverage has used Kerlink LoRa IoT station model 0X80400AC [41] and nodes designed by an Italian group start-up [42].

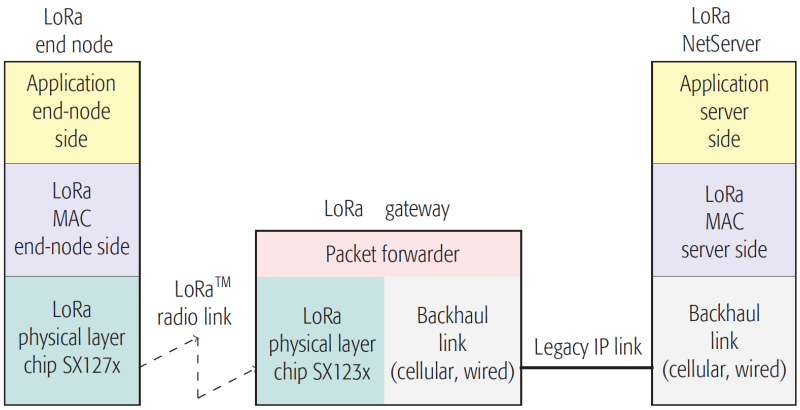



Figure 2.11: LoRa protocol architecture [23].

After several failed attempts using wireless and wired solutions, have been used LoRa in order to monitor and control temperature and humidity of different rooms in a 19 floors building installing 32 nodes (at least one per floor) and a gateway on roof top. It also has been made stress tests to challenge radio connectivity putting nodes inside elevators, for example, freaking out in a successful response.

Furthermore, has been carried out the coverage range test of LoRa network in Pandova, Italy. Inside an urban environment, the experiment has been assessed the worst technology coverage case, estimating the necessary gateways number to cover whole city.

### Relevant Topics

In general view, this research gives theoretical bases comparing communication technologies in LPWAN paradigm that use unlicensed bands. Even more, this project made a applied study where the results have been able to fortify why LoRa is a good option to implement an IoT system full of long-range communication benefits.

# Chapter 3

# Flow Control of Pedestrian Traffic

This chapter will present the used tools and the idea behind the project, at the hardware and software level, to develop a smart device able to perform the SSL system applicable on Crosswalks and also be integrated in a Smart City Environment.

## 3.1 Proposal of Solution

Such as described in Chapter 1, there are a general, and local, demand to prevent premature actions and give accessibility in pedestrian traffic. Therefore, it is necessary to develop a safe way to people cross the street on bad lighting places, also synchronizing both people and vehicles flow, seeking to protect the pedestrian.

This issue can be seen across problems with lighting quality on the crosswalk, the bad signalization for the driver and unpredictable pedestrian's actions during the crossing process. Bearing in mind these problems, this project propose a system compound by two Smart Devices, where they have the same schematic design with two similar embedded systems.

Each Smart Device is mainly compound, at the hardware level, for a Printed Circuit Board (PCB) controlled by a Raspberry Pi which uses LoRa, a Relays Module, a Voltage Regulator and, as energy source, it uses Photovoltaic Panels. At the software level, it is a program responsible to make the system control and propose a smart and secure solution

toward people flow on crosswalks.

The solution's physical scheme can be seen in Figure 3.1 and propose a system where are used a pair of different smart devices which communicate themselves Peer-to-Peer (P2P) using LoRa technology to makes a dynamic paired relation between them, synchronizing the pedestrians and vehicles flow.

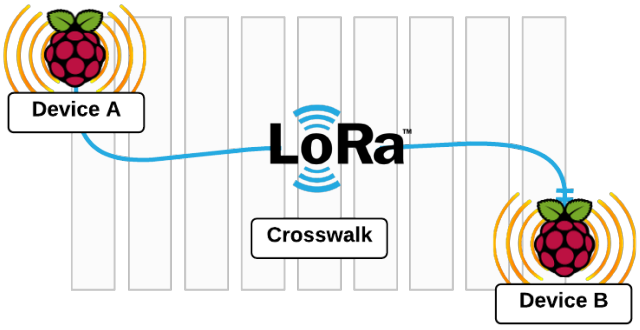



Figure 3.1: Smart Devices' pairs disposition on the crosswalk.

The system is idealized to work just during the night in cases where the pedestrians' traffic light are not connected with drivers' traffic light. In most of cases, the streets which have low or medium vehicles' flow and its does not need specific signalization to both. This work is projected to single way streets where the vehicles come, all the time, from the same direction and the amount of detectable pedestrians are considered as a single entity.

The differences between each device are shown in Table 3.1 and will be described in Topics 3.2.3 and 3.2.4. Both devices - A and B - has the peripherals quoted in Table 3.1 marked with the check signal. Making a comparison, the [*] highlights that cameras' capture areas are not the same, and blank spaces show that the Device B do not have the Pedestrian Light Board neither the Flashlight. These outputs are unnecessary because assuming that there is one sense's vehicle flow, only one device needs to signalize the crosswalk to the drivers.

The project explore the Lamp Unit idea that has been shown in Subsection 2.2.1. Thereby, the system use this structure to becomes able to detect pedestrian presence, ensure a well illuminated crosswalk and, focused in security and accessibility for pedestrians,

| Peripherals | | Device A | Device B |
|---|---|---|---|
| **IN** | Cameras* | ✓ | ✓ |
| | Pedestrian Detection | ✓ | ✓ |
| **OUT** | Focal Spotlight | ✓ | ✓ |
| | Crosswalk Light Board | ✓ | ✓ |
| | Flash Light | ✓ | |
| | Pedestrian Control Light Board | ✓ | ✓ |

Table 3.1: The input/ouput peripherals' difference between Device A and B.

signalize to them and to the drivers the exact moment to stop, wait and go ahead.

Figure 3.2 shows a scheme of the proposal idea to pedestrians' cross street process, which use as reference the paper [30], and the relevant detection areas inspired in the paper [36] as well as the some services that can be offered. Highlighted areas assigned with letter A or B correspond the type of device that is responsible to monitor them.

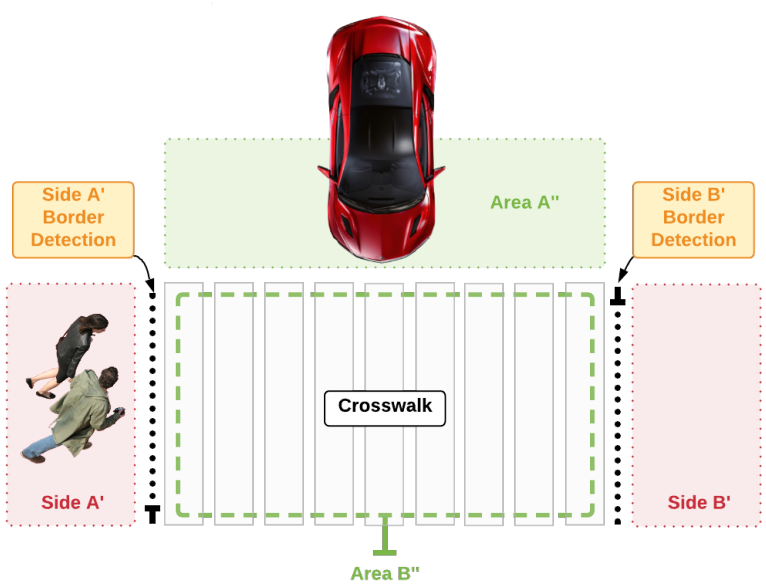



Figure 3.2: Pedestrians street cross scheme with relevant detection areas highlighted.

Furthermore, the street cross starts on side A' toward side B' using the crosswalk path (the same process can be make from the side B' toward side A'). Using the first case, it is considered that pedestrian stops before side A' border and stay on a place where the crosswalk's illumination activates when the pedestrian must to go ahead. So, the system analyses if there are vehicles getting close on the Area A" to ensure security during the

cross.

Thereafter, the system wait the exact moment where there is not cars or they are stopped on street, and it signalizes to pedestrian that he can cross safe toward side B'. At this moment, this pedestrian go ahead and pass through the side A' border detection activation lighting resources and is deactivated when the side B' border detection is triggered.

It is important to highlights that it is verified before finish the process if another pedestrian has crossed the side A' border or still there are any pedestrians on the area B". All this interaction between pedestrians and the system is shows in Figure 3.3, where the green block starts the process, the red one finishes, the purples are the actions make by pedestrians and, on the other hand, the blues and the oranges are, respectively, the actions and decision of the system.

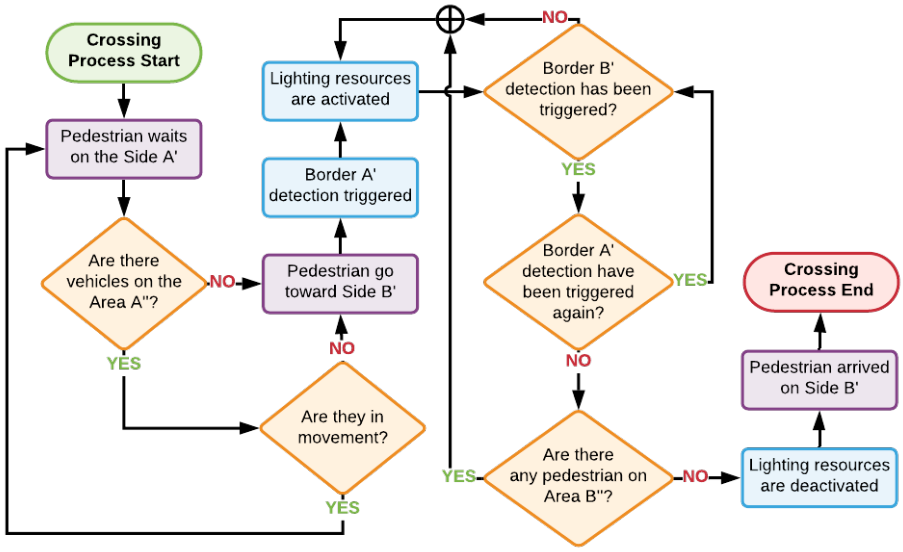



Figure 3.3: Pedestrian and System interaction process flowchart to cross the street.

Finally, the Section 3.2 and 3.3, respectively, will describe the specific considerations to design the schematic and the embedded system for each device, A and B, bearing in mind all the ideas discussed until now.

## 3.2 Hardware

### 3.2.1 Overview

The system's hardware for each device have the same schematic structure, being differentiated according have been seen in Section 3.1. The schematic electrical properties are based, in general, according the outputs, inputs and Integrated Circuits (IC) demands. About the energy supply, is considered that the Smart Device and its peripheral are supplied by hypothetical photo voltaic panels with its efficiency based on solar cycle of Bragança, Portugal.

Thereby, the schematic project must be able to represent the minimal features which the devices must perform. Since the circuit is not developed in practical environment, it is only explained their main modules relation and how its would work.

Bearing in mind these points, the development's methods are focused in theoretical schematic implementation. Finally, this Section describe all characteristics about the schematic and its Inputs/Outputs, explaining the method applied to develop them and all specific considerations for each peripheral. Figure 3.4 shows this project architecture organizing their inputs and outputs, while also shows the schematic diagram block.

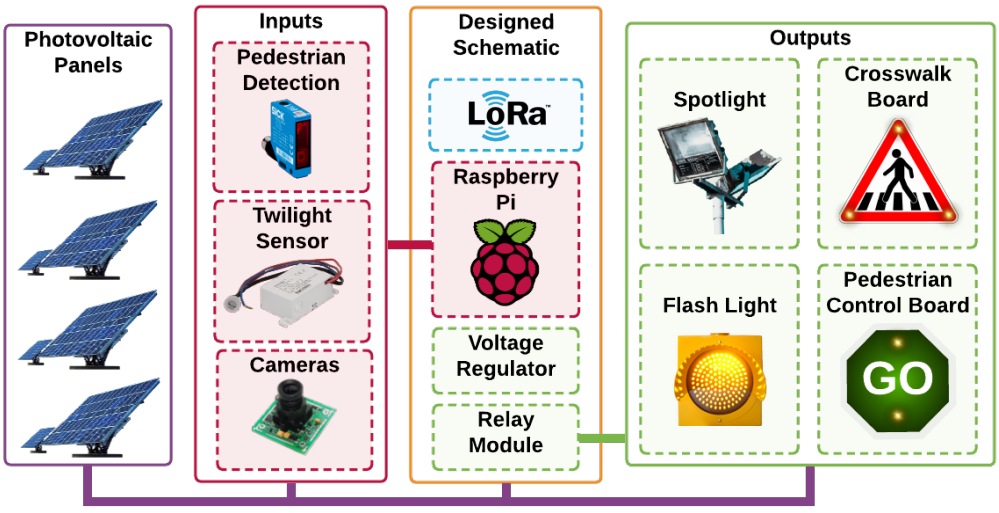



Figure 3.4: Hardware flowchart organizing schematic and its inputs and outputs.

As system's inputs are used a Obstacle Avoidance Sensor, an Twilight Sensor and a pair of Cameras for each device. As system's outputs system are used a group of light signals which are a Focal Spotlight, a Flash Light to the Drive, a Crosswalk Light Board and a Pedestrian Control Light Board.

### Methodology

This work does not encompass the structural project details, such as the pole's dimensions or accurate details about peripherals disposition. Furthermore, the detection, lighting and signalization devices are considered such as logical inputs and outputs with electrical power demands which uses a data channel to communicate with the control board.

All these assumptions are due the high complexity that encompass the development of a commercial circuit schematic‘s project and its embedded system, thus, the development's methods are focused in theoretical schematic implementation available on [43].

### KiCAD and Instructables

The schematic project is developed using an open-source software to prevent license problems and encourage project sharing in hobbyists and technology designers communities. The electronics' design community have been growing increasingly and, to give directions to this work it is used projects from Instructables, a web place to share projects.

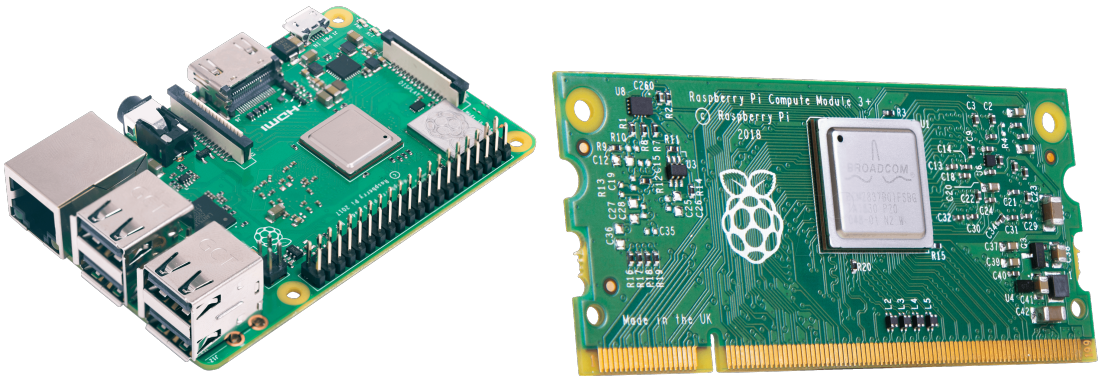
Several projects available on this platform are KiCad based, an open-source software for Electronic Design Automation (EDA) which gives tools to handle a schematic capture that generate appropriate standard output files adopted by PCB industries. Thus to have more details about this software and the projects community the link references area available, respectively, on [44] and [45].

### 3.2.2 Control Board

#### Raspberry Pi

This topic describe relevant details about this schematic project's brain. Thus in components' choice is considered the resources' variety to develop a system easily programmable, applicable in outdoor systems and, in general, able to perform a local powerful processing directed to image processing. The previous described demands are justified due the proposed system's architecture demands explained in Topic 3.2.1.

In this way, is developed a system Raspberry Pi based and, due the quoted demands, are used a Raspberry Pi CM3 (Figure 3.5b) to develop the schematic project and, to validate the embedded system work, is used a Raspberry Pi 3 Model B+ (Figure 3.5a).




(a) (b)

Figure 3.5: Raspberry Pi 3 Model B+ (3.5a) and Raspberry Pi Compute Module 3+ (3.5b).

The schematic design use as base an Instructables's project that is available in [46], and its only restriction it is not uses the Compute Module Lite version because this reference are idealized for the common version. Furthermore, this adaptation is not designed to has an structure to flash the CM3 module, thus, it is adopted that the system is previously flashed by an Compute Module IO board [47], following [48] as step-by-step installation.

According [49] this board contains the guts of a Raspberry Pi 3 Model B+ (the BCM2837 processor and 1GB RAM) as well as an eMMC Flash device with storage capacity defined, in this case, as 8GB which is equivalent to a SD Card slot on the Pi.

Both development devices' main features are available in [49], for the Compute Module, and [50], for the Single-Board Computer.

#### HopeRF RFM95W - LoRa Module

It is used the HopeRF RFM95W, that can be seen in Figure 3.6, which feature the LoRa$^{TM}$ long range modem that provides ultra-long range spread spectrum communication and high interference immunity with low energy consumption. All this device's features can be deep explored in [51], including the reference to chose the antenna type and the project's design parameters.

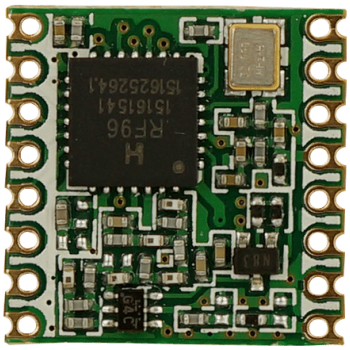


Figure 3.6: HopeRF RFM95W Module.

This module choice is motivated by the unlicensed communication band proposed through the LoRa technology which has been emphasized in the paper [23]. Thus, this module communicate with the Raspberry Pi using Serial Peripheral Interface (SPI) [52] and is applied to implement the wireless communication P2P between the devices A and B on the 868 MHz unlicensed European band.

### 3.2.3 Inputs

The inputs are shown in Figure 3.7 that schematizes their disposition and the coverage area according the letter, A or B, for each device's type.

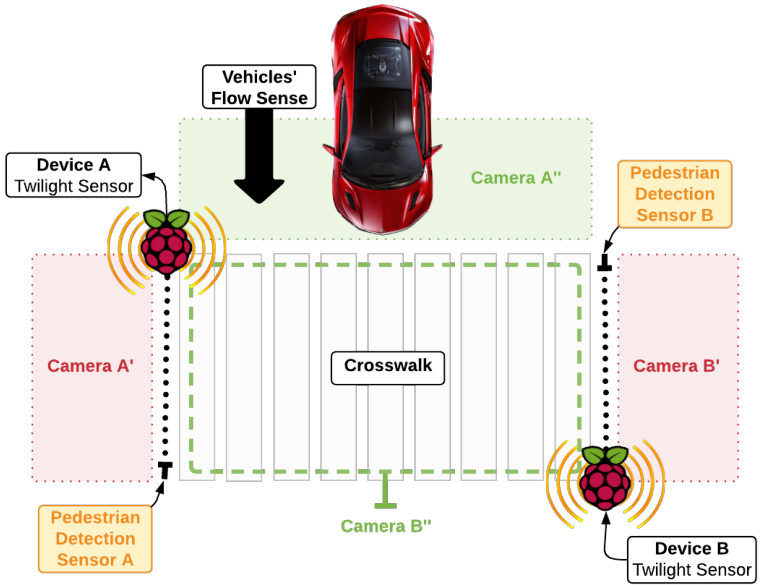



Figure 3.7: System inputs' disposition diagram.

The signal conditioning, signal's noise and sensors' accuracy are disregarded, it is only considered the power consumption. Thus, the input devices are assumed as a black box systems which acquire the environmental variations and gives to the system logic inputs as true or false. This Boolean values are read by a microprocessor responsible to decide the system work based on the embedded software described in Section 3.3

The matched devices make a system endowed for 8 inputs that create a structure able to detect if the system should start or finish its work as well to monitor the pedestrian and cars flow near the crosswalk. It is capable to identify pedestrians on the moment before entering the crosswalk in A' and B', on the crosswalk in B", and the cars that must wait during the pedestrian's crossing in A" according the Table 3.2.

| Sensor | Task |
|---|---|
| Twilight Sensor | System ON/OFF (Night/Day) |
| Camera A' and B' | Pedestrian before entering the crosswalk |
| Obstacle Avoidance | Pedestrian's entrance in crosswalk zone |
| Camera A" | Cars movement near crosswalk |
| Camera B" | Pedestrian crossing the street |

Table 3.2: Detection's task for all inputs.

First of all, the Twilight Sensor work as a switch which detects the night and it turns on the detection of people and cars near the crosswalk based on the crossing process proposed in Figure 3.3. The Cameras, in general, detect cars or people while the avoidance obstacle detects the exact moment that pedestrians enter on crosswalk.

### 3.2.4 Outputs

The outputs are shown in Figure 3.8 which exemplifies their disposition and it illustrates the idea for type A of devices. This diagram also shows the Spotlight focus coverage that, when associated with the device's type B on the other sidewalk, provides a complete crosswalk's lighting.

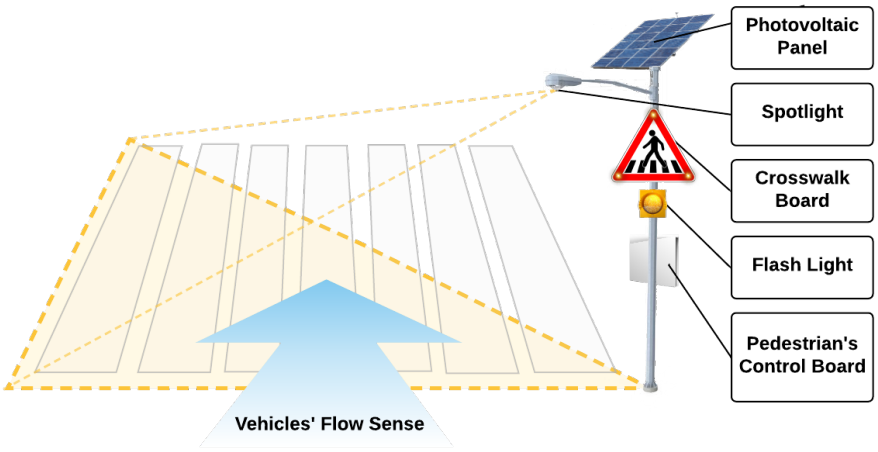



Figure 3.8: System outputs' disposition diagram for Device Type A.

The Crosswalk Board and the Flash Light are orientated face to the vehicles' flow sense while the Pedestrians' Control Board aim the sidewalk to alert them when they must enter on the crosswalk. Devices' type B do not need to has a flash light because is assumed that cars come from an only sense. The outputs have they tasks according the Table 3.3.

The Crosswalk Light Board is activated at night while the another outputs just are activated during the crossing process. As soon as the pedestrian is getting close of crosswalk through the sidewalk, the Flash Light is activated.

| Lighting Feature | Task |
|---|---|
| Focal Spotlight | Light the crosswalk |
| Crosswalk Light Board | Signalize the crosswalk |
| Flash Light | Alert the driver of a crossing |
| Pedestrians Control Light Board | Alert the pedestrian to go ahead |

Table 3.3: Signalization's task for all outputs.

The Pedestrians Control Light Board is activated when the software allows the crossing as well the flash just stop to blink when there are not more pedestrians to cross the street. Finally, when the pedestrian definitely enter on the crosswalk, the Focal Spotlight lighten the crosswalk.

### 3.2.5 Energy Supply

The systems' energy demands are projected considering a energetically autonomous, that is, all the electric consumption must be supplied by this photovoltaic panel. Grossly, this panel is considered based in solar radiance variation in Bragança, Portugal.

Idealizing a real place to use as reference, it is used a crosswalk near to Praça do Prof. Cavaleiro de Ferreira (Lat/Lon:41.807/-6.759), shown in Figure 3.9a, and its monthly solar radiation data during 2016, in Table 3.4. The data is provided by an European Commission tool [53] which has as input to get the data this crosswalk's latitude/longitude and the inputs shown in Figure 3.9b.

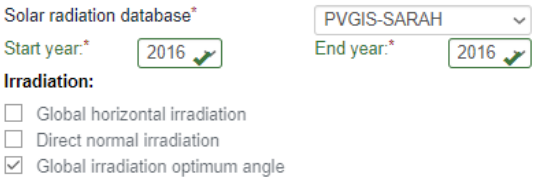
Therefore, using this real place as example, it is also necessary to use a real panel (brand, efficiency and dimension) to estimate the system energy restrictions. In that way, it is used a REC group panel's model [55] (the largest solar panel European brand [56]), which has as dimension D = 1.67 $m^2$ and Efficiency $\epsilon = 18\%$ .

| **Jan*** | **Feb** | **Mar** | **Apr** | **May** | **Jun** | **Jul** | **Aug** | **Sept** | **Octo** | **Nov** | **Dec** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 53.1 | 58 | 120.3 | 137.6 | 156.7 | 203.6 | 243.8 | 226.7 | 185 | 138.7 | 80.2 | 101 |

Table 3.4: 2016 Monthly Radiation (KWh/$m^2$).

Thus, in Table 3.4 is possible to see that January is the month with the worst solar

(a)




(b)

Figure 3.9: Example crosswalk [54] near IPB (Lat/Lon:41.807/-6.759) (3.9a) and inputs of European Commission Tool (3.9b).

energy ($P_{jan}$) production in Bragança, in this ways, it is estimate the system power restrictions using this value and the previously quoted REC panel's characteristics applied in Equation 3.1.

$$P = D\epsilon P_{jan} \tag{3.1}$$

Where, P in [W] is the system power restriction for the REC panel in January. To estimate the system consumption is used as reference that the system work during all the night and each output is activated until 20 seconds 100 times/night. Each work cycle's power consumption will be considered as the power demands' sum of the control board and of inputs/outputs listed in the Table 3.5.

All the consumption estimates are based on Equation 3.2. where the electrical variables, current and voltage, are analysed on cases assuming the highest power demands. This values can be seen in each devices' datasheet available in the Table 3.5.

$$P_{con} = V_{ss} I_{max} t \tag{3.2}$$

Where $P_{con}$ is the Power Consumption in [Wh], $V_{ss}$ is the supply source voltage [V], $I_{max}$ the major current demand [A] and $t$ is the work time [h] per month during the night. The all system consumption can be evaluated in the Equation 3.3.

| | Device | Work Time (h/month) | Power Demand (W) | Power Consumption (Wh) |
|---|---|---|---|---|
| **Control Board** | Microprocessor [49] | 744 | 3.5 | 2604 |
| | Communication [51] | | 0.4 | 298 |
| **IN** | Cameras [57] | 372 | 2.5 | 930 |
| | Obstacle Avoidance [58] | | 0.125 | 46.5 |
| | Twilight Sensor [59] | - | - | - |
| **OUT** | Crosswalk Board [60] | 372 | 1 | 372 |
| | Flash Light [61] | 13 | 9 | 117 |
| | Spotlight [62] | | 26 | 338 |
| | Pedestrian Board [60] | | 1 | 13 |

Table 3.5: Power consumption estimate for devices' reference.

$$P_{con} = \sum P_{cb} + \sum P_{in} + \sum P_{out} \tag{3.3}$$

Where $P_{con}$ is the sum of all system peripherals including the controal board ($P_{cb}$), inputs ($P_{in}$) and outputs ($P_{out}$) all measured in [Wh].

## 3.3 Software

### 3.3.1 Overview

This embedded system is Python based and use Object-Oriented Programming (OOP) concepts to implement the algorithms. The main used concepts dive in the idea of classes, attributes and methods [63].

The algorithm creates 2 classes where one of it represent the Raspberry Pi and another the LoRa communication module. It also uses some functions' module and .csv files to operate the system logic anf the Figure 3.10 represents the relation between them.

The Raspberry Pi class has some of attributes get from the files "device.csv" and "authtorizedDevices.csv" which are responsible to define the device's characteristics, for example, its type, A or B, and an unique ID.

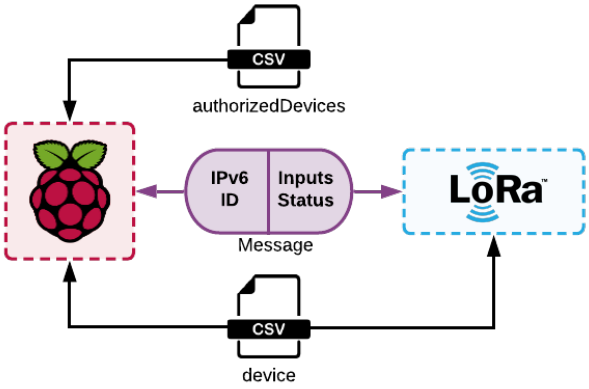



Figure 3.10: Main algorithm's scheme.

The LoRa class also define attributes using the "device.csv", in this case, the communication channel. Each one have their methods defined based on the tasks that must be performed. More details about these attributes and methods will be shown in Chapter 4.

### 3.3.2 Methodology

The main algorithm is able to implement the communication between the devices A and B, read the inputs variations, activates the outputs and, finally, update the device pair in real-time. This update send a specific message package compound by th IPv6 unique ID and the command logic's array.

The system works in infinity looping reading/writing the the inputs and outputs as boolean values. It releases an inputs status' array , makes output decisions and communicates their pair about new inputs variations. Figure 3.11 shows that the communication works in constant RX mode and just uses the TX mode to send a status update when an input variation is detected.

The inputs are read constantly and they are updated on the status' logic array. This is sent to device pair providing a pared decision making, that is, both devices work synchronized. All sent messages are encrypted and only are accepted if the unique ID attached on it exist inside the receiver "authorizedDevices.csv" file.

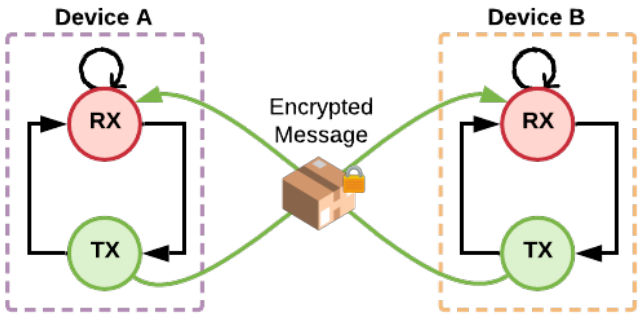



Figure 3.11: RX/TX communication method.

### 3.3.3 Inputs and Outputs

The Inputs and outputs for devices are treated as Rasbperry Pi class' methods and are activated in sequence according the inputs detection. This relation between detection and activation is described in Figure 3.12


Figure 3.12: Inputs and Outputs' work timeline.

This Work Timeline starts when the Twilight Sensor, working as a switch, connect the system with the battery bank as soon as the dusk is detected. The Crosswalk Board starts to work right after this while the Raspberry Pi initialize their operating system. The Twilight Sensor and the Crosswalk Board still on during all night.

The algorithm wait Cameras detection on areas A' and B' to turn on the Flash light which alert the drivers that the crossing process will start. Another Camera verify if there is movement on area A", just in case that there is not movement, is activated the

Pedestrian Control Board.

The Spotlight only is activated when the pedestrian, in fact, enter on the crosswalk and he activates the Obstacle Avoidance Sensor. In the end, the system waits until there is not more pedestrian on area B" to turn off all outputs and restart the detection process.

### 3.3.4 Development Tools

The computational tools employed to develop tests and code the embedded system are in the Table 3.6 with their respective tasks.

| **Tool** | **Task** |
|---|---|
| Putty | SSH Remote Access |
| Angry IP Scanner | Identify Microprocessor IP |
| Raspbian | Operating System |
| Python 3.7 | Programming Language |
| VNC Viewer | Remote Access' Visual Interface |

Table 3.6: Computational useful tools.

This tools are used to construct the programming environment. As well the system are a Python code that must be embedded on a Raspberry Pi, it is convenient make test with direct access of microcontroller functionalities.

In this ways are used an Operating System called Raspbian [64] and coded the system in Python 3.7 [65]. The Angry IP Scanner [66] is used to detect the Raspberry Pi IP which should be used on Putty [67] and on VNC Viewer [68] to make the remote connection from a personal computer to microcontroller.

# Chapter 4

# Development: Embedded System and Schematic

The Schematic Development is explained in five main blocks: LoRa RFM95W, inputs connectors, Raspberry Pi Compute Module, Output relays and Energy Supply. In all of them will be explained the considerations to develop this hardware structure and, in the end, was made a power consumption's analyse that will be shown results in the Chapter 5.

The Embedded System Development is explained in six main parts: Work Environment, Pilot, Structural, Communication, Integration and System Incorporation. These Subsection encompass what is used to develop the software including useful programs, components to assembly the tests and the versioning progress to make Final Embedded System.

## 4.1 Hardware

The circuit proposal, in Figure 4.1, has been developed using the Eeschema, an schematic editor, yielded by the KiCAD platform. It has been based on design criteria from datasheets and from the base project available on [46].

The next Subsections will explain technical details for each of these blocks and, when

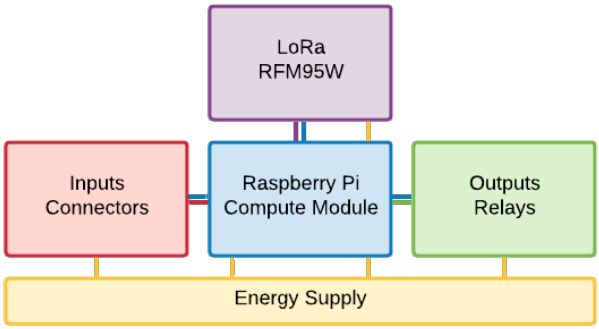



Figure 4.1: Block diagram of schematic's main components.

necessary, it will traces a parallel with the base project explaining the necessary modifications. The schematics used as reference are all attached in Appendix A.

### 4.1.1 Raspberry Pi

The Raspberry Pi schematic is the project core and provide the connection between all other features available on the circuit. Therefore, as well described by the [46] author, the base project goal have been a board project able to provides a kick off design to Raspberry Pi based products, and it has the schematic sheets in the Appendix A.

This kick off design is the perfect fit to start the Smart Crosswalk circuit, however, some details have been adapted due the specific project demands. The original project has generic features that have been removed. Some connectors type such as HDMI, GPIO header, USB, Micro-USB and Female Audio Jack are unnecessary once that this work project is a circuit applicable on outdoor environment wired in particular peripherals.

In this way, after the system incorporation, using the Compute Module IO board, the project must work autonomously, and it is only removed of the smart device for maintenance. All the made adaptations are in the Appendix B.

After theses considerations, the first step to develop this circuit have been consult [69] to understand the restrictions that should be adopted to modify the base project. The Raspberry Pi CM3 is a wired in a 200 pin DDR2 SODIMM connector, thus, it has been necessary consult function of each pin.

These pins encompass all the Raspberry Pi functions such as the power supply and

peripherals. The energy supply pins are still the same while was manipulated just the GPIO pins. The GPIOS pins are divided in two banks which are represented for one block on the schematic, shown in Figure 4.2, that indicates the pins and their respective GPIOs.

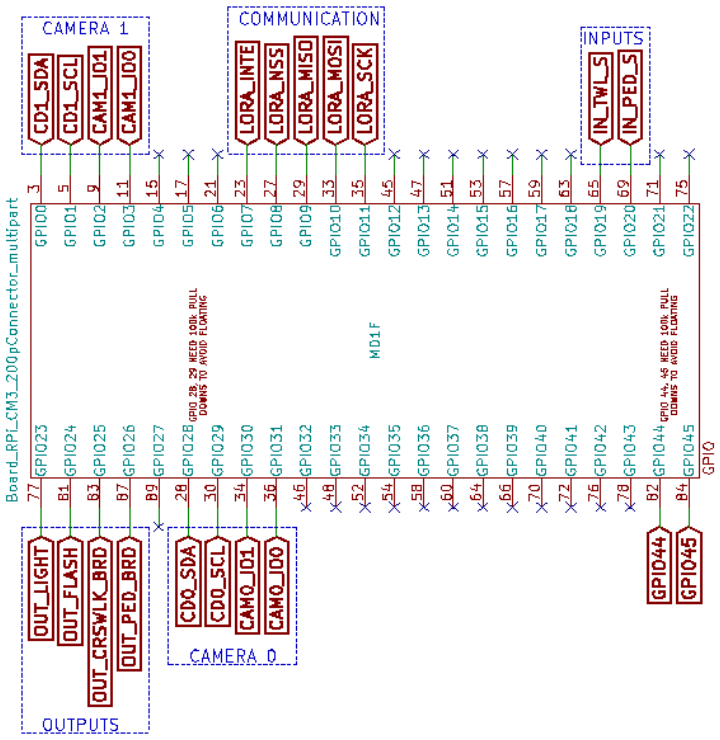



Figure 4.2: Block diagram GPIO banks.

Highlighted in blue blocks are shown the pins which connect the Raspiberry Pi CM3 with the communication module, the inputs/outputs and the cameras. Inside these blocks are the pins self explanatory labels, which indicate their functions and define where they are wired. The GPIO44 and 45 are wired in 100KΩ pull-down resistors, according advised by the design manual, to avoid floating values.

In the Appendix B, Figure B.1, is shown all pin's connection blocks besides a capacitors net and a MOSFET circuit, shown as a cut in Figure 4.3. Both of them are proposed by the base project and do not suffer changes.

The capacitors are used to avoid noises and prevent damages on sensible components of the circuit. The Compute Module has a pin called EMMC_DISABLE_N which when

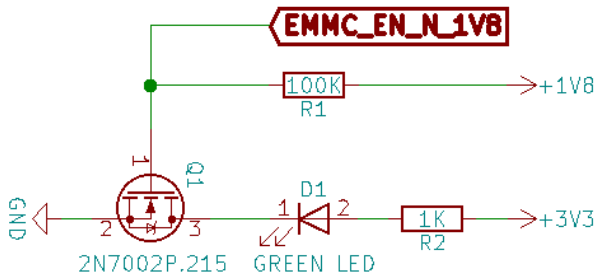



Figure 4.3: Re-enable SD/eMMC MOSFET circuit.

shorted to GND will disable the SD/eMMC interface to force the BCM2837 to boot from USB, which is necessary to flash the memory with an Operating System.

After the flash process, any board CM3 based need to use the Figure 4.3 circuit to re-enable the eMMC device to allow access to it as mass storage and, finally, make the system works properly.

### 4.1.2 LoRa RFM95W

The communication module is based on HopeRF RFM95W as described in Subsection 3.2.2. This module uses SPI to work together the CM3 and use the pins connection as shown in Figure 4.4 which are wired according the Table 4.1.

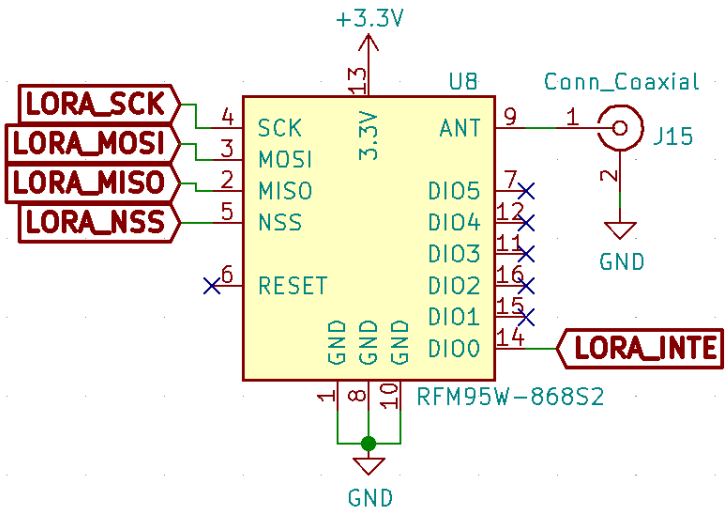



Figure 4.4: HopeRF RFM95W communication module.

The labels show the connections make with the CM3 and each one have an specific function. The pin 9 is the pin where the antenna of the module must be connected. Because this, it has been put a coaxial connector where can be plugged an antenna out of the board.

| Compute Module | GPIO | RFM95W | Pins |
|---|---|---|---|
| SPI0_CE0_N | 8 | NSS | 5 |
| SPI0_MISO | 9 | MISO | 2 |
| SPI0_MOSI | 10 | MOSI | 3 |
| SPI0_SCLK | 11 | SCK | 4 |

Table 4.1: SPI connection between LoRa and Raspberry CM3.

The pins 2, 3, 4 and 5 are responsible to ensure the SPI interface and they are connected according the datasheet available on [51]. The pin 14 uses the GPIO7 which is set such as interruption when some message is received.

### 4.1.3 Inputs Connectors

Basically, these connectors have been destined for two inputs types: Cameras with flex flat and sensors attached on a 3-way screw terminal. These peripherals model have been described in Subsection 3.2.5, Table 3.5 and each one use specific connectors.

The cameras connector was modified when compared with the base project. Figure 4.5 show the old model, in the left, and the adapted model, in the right. The old model uses a 22-way flex flat connector which needs to use a adaptor to connect in a 15-way flex flat, which is the camera's cable type. To eliminate this adaptor demands was used a 15-way connector instead of the 22-way model. The specific pins and GPIOs can be seen in Appendix B in Figure B.1 and where they are wired in Figure B.3.

The Obstacle Avoidance Sensor and the Twilight Sensor are attached on a 3-way screw terminal as can be seen in Figure 4.6. In both cases these connectors have 2 pins wired on power supply (5V/3.3V and GND) while the last one is the input values' channel.

The Obstacle Avoidance (GPIO20) is supplied for 5V while the Twilight Sensor (GPIO19)

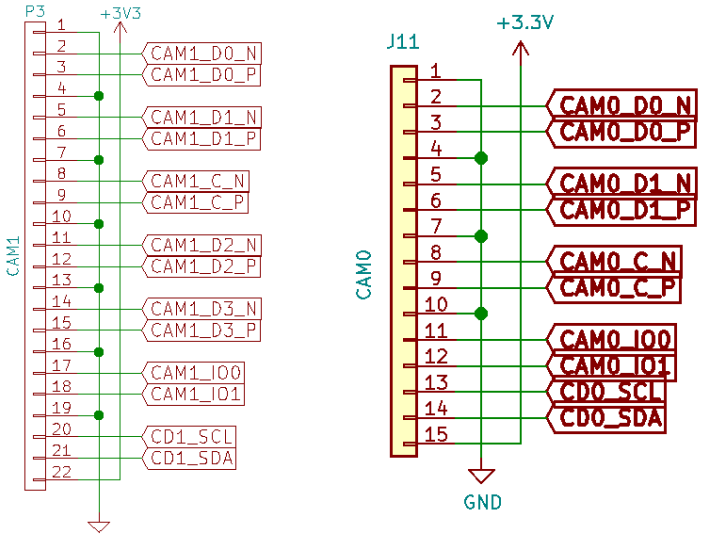



Figure 4.5: Improvement of the flex flat connector from 22-way, in the left, to 15-way connector, in the right.

for 3.3V. The first sensor work with a independent logic circuit while the second just work as a photocell switch, providing a logic value (HIGH as 3.3V) in a GPIO when activated.

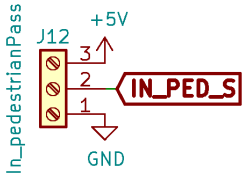



Figure 4.6: 3-Way Screw Terminal example.

### 4.1.4 Outputs Relays

All the idealized outputs are activated using, by default, the relay's drive circuit seen in Figure 4.7. This circuit is responsible to turn on the outputs with 12V and provide current direct from the batteries. The relays are 12V/3A and they drive the output for a HIGH value when the respective GPIO provides a LOW value.

The Table 4.2 shows which output are activated for each GPIO. The label wired in the Optocoupler's pin 2 connect the circuit with the CM3 and defines when the output will be activated. When it is a LOW value, the NPN transistor is triggered, receiving a

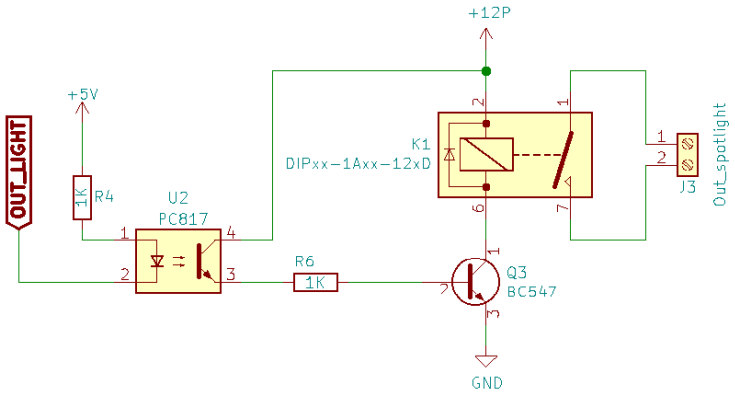



Figure 4.7: Generic relay circuit used on all outputs.

base current, which makes the relay conduct and, finally, it actives the output.

| Output | GPIO |
|---|---|
| Spotlight | 23 |
| Flash Light | 24 |
| Crosswalk Board | 25 |
| Pedestrian Control Board | 26 |

Table 4.2: Outputs wiring.

Without this circuit would be impossible to activate the outputs only with the electrical power provided by the control board. It is broadly used on several electronic projects where exist an output with electrical demands larger than the command board circuit can provide, which it is the case.

### 4.1.5 Energy Supply

The circuit has 3 different voltage values and all of them are provided by voltage regulators. The main regulator, shown in Figure 4.8, is responsible to take 12V from the battery supplied by the Photovoltaic Panels and Step-Down this value in 5V, voltage which the CM3 works.

The conversion process starts with an screw terminal where is attached the batteries which is is charged with energy from the photovoltaic panel. In the sequence, all the

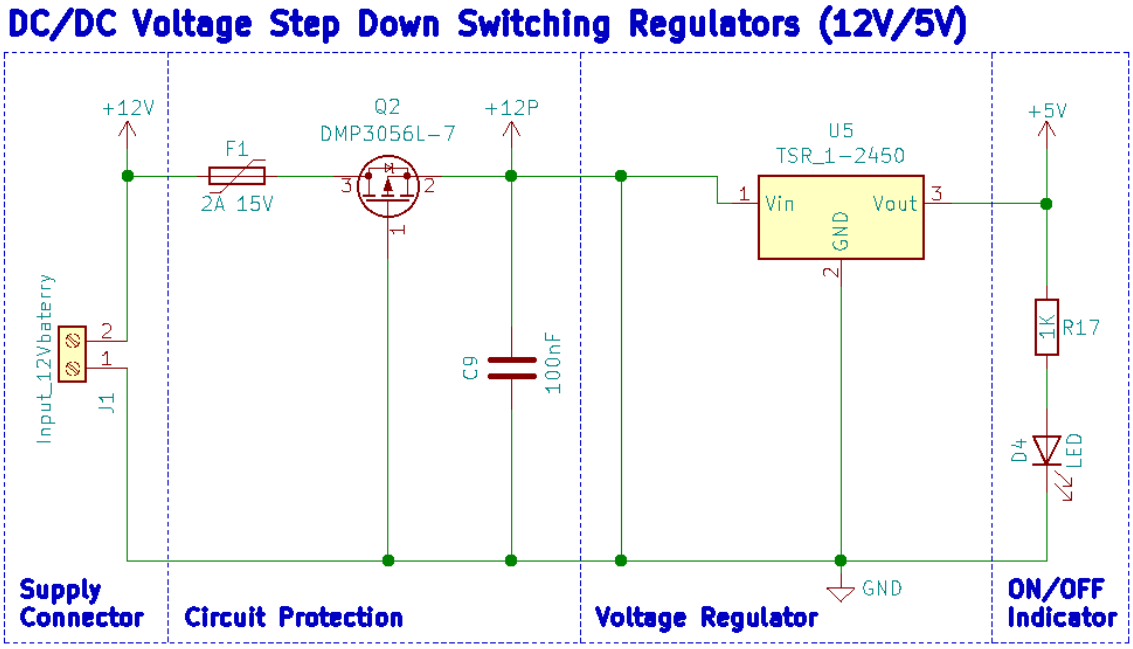



Figure 4.8: DC/DC Voltage Step Down Switching Regulators (12V/5V).

control circuit is protected by a fuse and a MOSFET that minimizes the on-state resistance improving the switching performance.

The regulator step-down 12V in 5V and endure 1A, and when is activated turn on a LED which indicates that the circuit is ON. The Compute Module also uses 3.3V and 1.8V which are provided by the same regulators family, shows in Figure 4.9, and they endure, respectively, 0.8A and 1A.

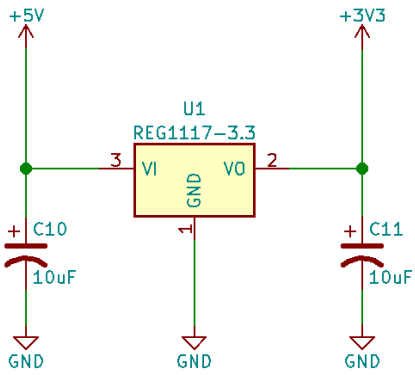



Figure 4.9: DC/DC Voltage Step Down Switching Regulators (5V/3.3V or 5V/1.8V).

## 4.2 Software

The proposal solution, in Section 3.1, implement a pattern to crossing the street on crosswalks and ensure the pedestrian safety within lighting features. Thus, understand the idea seen in Subsection 3.3.2, the software needs to simulate a system which read 4 inputs and, through them, be able to control the flow on crosswalk using 4 lighting outputs.

This crossing scenery have been simulated using the physical components' list shows in Table 4.3 and the tools quoted in Subsection 3.3.4. These components show if the algorithm process the inputs correctly then it reacts with the outputs on the correct circumstances.

| Amount | Components |
|---|---|
| 2 | Raspberry Pi 3B+ (with Power Supply) |
| 2 | LoRa Module RFM95W 868 MHz |
| 2 | Breadboard (400 Holes) |
| 2 | Push-button |
| 8 | 330Ω, 10KΩ and 100KΩ Resistors |
| 8 | Led (with Different Colors) |
| 20 | Male-Male Jumper |
| 40 | Female-Male Jumper |

Table 4.3: Necessary components to simulate the algorithms.

The algorithms have been incrementally updated until this final version. This versioning progress follows the Table 4.4. Each version had principal changes focused on the main program structure, the communication process and how the programming language's tools were used.

| Version | Focus |
|---|---|
| Pilot | Technologies familiarization |
| Structural | Implement two different types of Smart Crosswalk |
| Communication | Method of RX continuous/TX to send based in a Flag |
| Integration | Join the Structutral and the Communication implementations |

Table 4.4: Developed algorithms' versioning.

Bearing in mind the problem interpretation and how to simulate it, first will be explained the Work Environment and after this the Implementations Versioning around five subjects: Goals, Approach, Program Work and the Challenges and Solutions.

### 4.2.1 Work Environment

The practical process include some steps, which are shown below, to elaborate the work environment with the quoted tools that have seen in Chapter 3, Subsection 3.3.4. The process can be briefly described as to prepare a Raspberry Pi 3B+ with an operating system to make tests with Python scripts.

1. Install the Raspbian on the Raspberry Pi's SD card.
2. Access this microprocessor using SSH as remote access.
3. Use the Python's development environment.
4. Make tests in circuits assembled on a protoboard.

The operating system installing are based on the Raspberry Pi community guide [70], and can be accessed using remote communication or attaching the board in a keyboard and a screen. After the installation, the first boot have been made using remote communication as follows:

- Attach the microprocessor board in a domestic network using a Ethernet cable plugged in a internet modem.
- Use the Angry IP Scanner [66] to discovery the device IP that will be accessed remotely. Figure 4.10 shows the software window and how look like the Raspberry Pi label.
- With the found IP open the Putty [67] window (Figure 4.11) and access remotely the microprocessor.

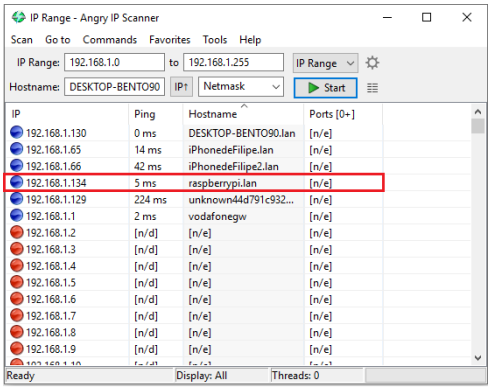



Figure 4.10: Interface to found the Raspberry Pi IP on Angry IP Scanner window.

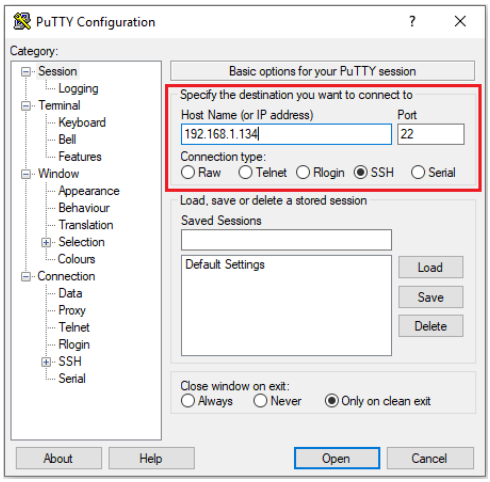



Figure 4.11: Access Putty window to connect remotely with the Raspberry Pi.

- Now, the remote access looks like a Linux terminal (Figure 4.12), and request user and password. The Raspbian has a default user name *pi* with the password *raspberry* and, after these inputs, it is accessed the Raspberry configurations with the command **raspi-config**.

- It is enabled the VNC in the option *Interfacing Options* (Figure 4.13 to provides access using a graphical interface.

- Finally, using the VNC Viewer [68], it is possible to use the Python 3.7 environment (Figure 4.14) on the Raspbian desktop.

The IP seen in Figure 4.10 variate every new network and if it is used Ethernet Cable

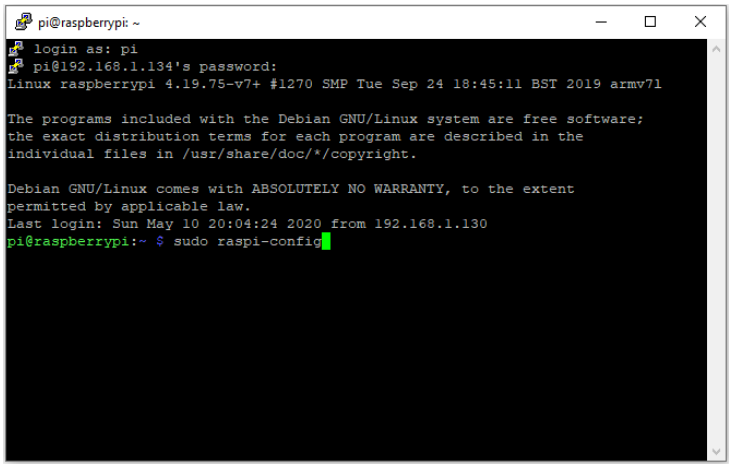



Figure 4.12: Remote access terminal screen.

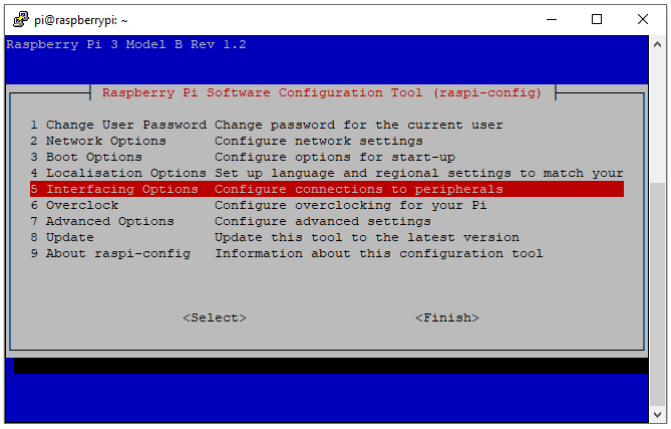



Figure 4.13: VNC enabling on *Interfacing Options*.

or Wi-Fi connection, because this, this IP it is not default and can be different. Inside this environment have been implemented codes tested using the the circuits that will be described on the follow Subsections.

### 4.2.2 Pilot Implementation

The pilot implementation has been the first contact with the development tools Furthermore, the idea behind this pilot algorithm has gave this work's directions and shown some challenges that would appears in the future.

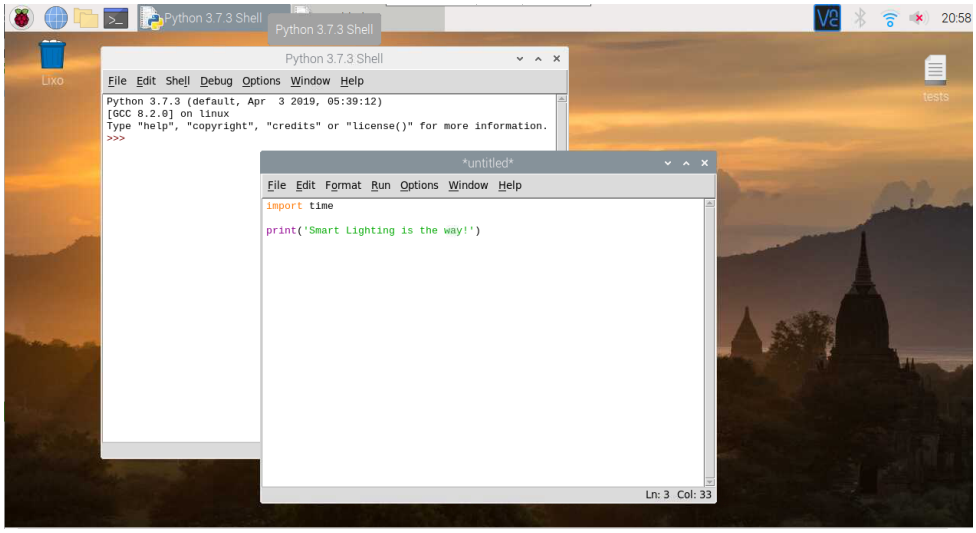



Figure 4.14: Python 3.7 environment on the Raspbian desktop.

### Goals

Therefore, it is shown its full code on Appendix C.1 and this will be analysed each part of their codes trying to show the solutions that have been found to achieve the following goals:

- Familiarization with Raspbian environment with Python 3.8 scripting
- Wiring the communication module RFM95W on Raspberry Pi 3B+
- Define a communication protocol between the devices
- Develop a code structure to emulate a simplify version of the Crossing Proposal seen in the Section 3.1, Figure 3.3.

The adopted simplification is for only the two main inputs and all the outputs for both devices, although that the device A is different of the B, such as defined on the Subsection 3.1, Chapter 3.

Following these three goals, the first step has been prepare the Raspbian environment such as explained in the Section 4.2.1. After this preset, the development process has started and was determined an Approach.

### Approach

The first step was understand how the communication module work together the Raspberry Pi. In this way, following the RFM95W datasheet, has been figured out that they must be wired using SPI such as shown in Figure 4.15.

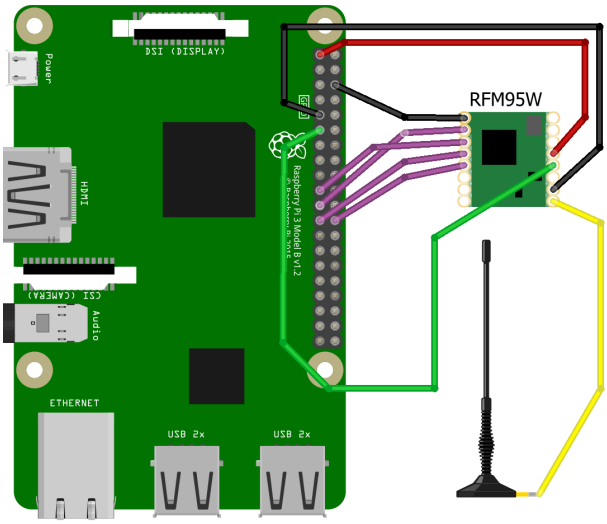



Figure 4.15: RFM95W and Raspberry 3B+ wiring.

This scheme shows all the connections between the Raspberry and the RFM95W according the Table 4.5, Wired in purple, the pins responsible for the SPI communication, in green, the responsible to activate the interruption routine on the Raspberry Pi, in red and black, the power supply (3.3V) and the ground (GND) and, in yellow, the antenna is attached on the module. The Appendix D.1, shows this real protoboard's assembly.

| **Function** | **GND** | **MISO** | **MOSI** | **SCK** | **NSS** | **DIO0** | **3.3V** | **GND** | **ANT** |
|---|---|---|---|---|---|---|---|---|---|
| **RFM95W** | 1 | 2 | 3 | 4 | 5 | 14 | 13 | 10 | 9 |
| **Raspberry** | GND | GPIO9 | GPIO10 | GPIO11 | GPIO8 | GPIO7 | 3.3V | GND | - |

Table 4.5: Connection between Raspberry Pi 3B+ and RFM95W.

The inputs and outputs are represented for, respectively 2 push-buttons and 4 LEDs, wired according the Figure 4.16 and the Table 4.6. The push-buttons are wired in a voltage divider compound by $R_1 = 100$ KΩ and $R_2 = 10$ KΩ. The GPIO is connected

between these two resistor while they ensure LOW value when the push-buttons are not hold, and HIGH value when this happens.

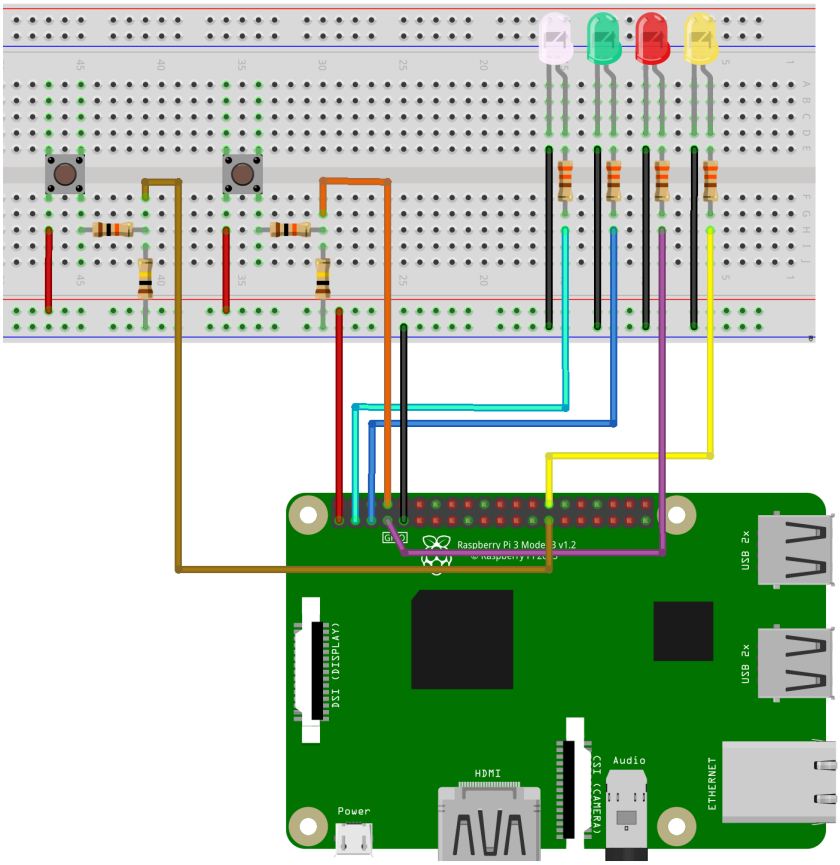



Figure 4.16: Pilot assembly.

| Inputs | | Outputs | |
|---|---|---|---|
| Twilight Sensor | GPIO14 | Spotlight (White) | GPIO2 |
| Camera 0 | - | Pedestrian Board (Green) | GPIO3 |
| Camera 1 | - | Flash Light (Red) | GPIO4 |
| Obstacle Sensor | GPIO0 | Crosswalk Board (Yellow) | GPIO1 |

Table 4.6: GPIO pins used as Inputs/Outputs for the Pilot version.

The LEDs are wired in a brunch with R = 330 Ω on its anode and GND on catode. The GPIO is connected in the other resistor side and gives HIGH value to activate the LED.

### Program Work

The program uses a non-default python's library called **raspi-lora** [71] and default libraries such as **GPIOZero** [72], **time** and **csv**. In general, they are used to develop all the algorithms' versions and can be briefly described as follows:

- **raspi-lora**: Provide tools to uses the LoRa RFM95W attached on the Raspberry Pi. It gives methods able to send messages, change the operating status (RX, TX and IDLE) and cryptography messages.
- **GPIOZero**: It is used to control the GPIO pins Raspberry Pi.
- **time**: Useful to manipulate time and generate delays.
- **csv**: Used to manipulate .csv files.

Using these libraries as tools, the program emulates a simplified scenery where the pedestrian wants cross the street and the system is enabled when he enters on the crosswalk.

The system is activated at night , enabling the Crosswalk Board, and activate the Spotlight, the Flash Light and the Pedestrian Board along a specified crossing time $t$ when the pedestrian is detected for by Obstacle Avoidance Sensor.

The system works is based on a status array which contains flags (True/False) to monitor the inputsoutput. In this program this status array is defined such as a list with 5 positions, [System, Flash Light, Pedestrian Board, Spotlight, Senor], with index between [0,4].

The device need to detect pedestrian in both sides of the street, because this, it has been developed an algorithm which would be embedded. In this case, it is installed in two equal devices and each one has their own unique attributes which enable them to work in pair, activating the outputs synchronously.

This attributes are an unique ID, the communication channel (a integer number between [0,255], the address that will be installed the devices and, in the end, the crossing time. An example of these attributes is shown on the line 26, in the Appendix C.1.1.

Each device has a .csv file called *devicesID.cdv* which lists another devices authorized to give output's enabling commands. The algorithm reads this file and creates a variable compound by an unique ID and the communication channel which must be accessed to send commands to the another device, starting the communication between the pair of devices.

This communication happens when, after the system detects the night, one of the devices detect a pedestrian crossing. When occurs, it is sent a package with a unique ID ,a command and a package ID to enable the outputs. This package ID count the amount of packages sent and is used to identify if the received package is new or not.

The device on another side of the street receive the package and only executes the command if the attached unique ID exist in the file *devicesID.csv*. This protocol ensure safety against message interceptions in a Smart City environment for example.

The way to represents it in code's lines, in first touching, has been creating a main script with the logical choices about the system's work and the decisions based on the interaction system/pedestrian. The main script uses three modules:

- **functions**: Responsible to provides functions to enable/disable the outputs, defines the GPIO such as inputs or outputs and code and decode packages that will be sent to the other device.

- **raspi-lora**: This module, in this case without changes, is provided by the library **raspi-lora** and has the functions already explained.

- **registers**: It includes functions which are used to read the *devicesID.csv* and adjust the authorized devices' information inside this file. It removes unwanted characters and ensure that they will read in a correct variable's type.

The main script were written seeking be self explanatory to enable another people to use it as program base and will be explained in lines' interval as follow:

1. Starts calling the modules and libraries (1-11).

2. Setup GPIOs which will be used as input/output lines (12-22).

3. Creates a tuple with the device' attributes read in the *devicesID.csv* and creates an *lora* object able to perform the remote communication (23-31).

4. Ensure that all outputs start deactivated, none status is True and none package habe been received (32-38).

5. Starts the main program which evaluate the inputs and activates the outputs according the pedestrian detection every time updating the status array (39-95).

    - Starts a constant looping and waits the night be detected for the Twilight Sensor, activating the Crosswalk Board when it happens (39-49).
    - Verify if a new package has been received and if the device which sent it is authorized. After this, it decodes the package and updates the messages received based on this integer number (50-64).
    - If the pedestrian is detected for the Obstacle Senor or a new package is received the outputs are activated during the crossing time. This time $t$ can be find in the tuple *SMARTCROSSWALK*, index 4 (65-85).
    - Verify in all main routine cycles if the Day already starts and the system must be deactivated (86-95).

**Challenges and Solutions**

The challenges to develop this version occurs bearing in mind the first contact with the technologies and how they working together can implement a smart crosswalk. Inside it, the main problem was how to wire the SPI to connect the Raspberry Pi and the RFM95W and identify if the received package is new or not.

The solution implemented to identify new packages, after some tests, generated problem. The integer number is incremented separated for each device, due this, they lose the synchrony when, supposedly, more than one pedestrian comes from the same size.

This problem has been solved understanding better the **raspi-lora** library and is reinterpreted in the next Subsections. This new versions will use new programming structures which were developed separately in three implementations: Structural (Subsection 4.2.3), Communication (Subsection 4.2.4 and, finally, Integration (Subsection 4.2.5).

### 4.2.3 Structural

Following the same idea proposed in the Subsection 4.2.2, the program gain some structural improvements. This version is developed for each device type, A or B, encompassing their inputs/outputs working for each type. All the implementations, including this one, use the same libraries and work inside the same crossing proposal.

#### Goals

Therefore, the full algorithm is shown on Appendix 4.2.3 and this will be analysed using the same Topics of Subsection 4.2.2. Each part of their codes try to show the solutions that have been found to achieve the following goals:

- Implement a algorithm based on Object classes and Methods.
- Develop a specific main algorithm for each device type, A and B.

The next topic describes some differences between the Pilot and the new version, mainly, focused in the new programming concepts used.

#### Approach

Now, the algorithm work with two different objects, the *LoRa* and *RaspberryPi* despite modules. In this way this Subection is focused in how has been created the Object class *RaspberryPi*.

This class has methods to command the inputs/outputs and attributes to store the device's unique characteristics shows in the Appendix C.2.2. The system algorithms for

each device type have little differences between them and is shows in the Appendix C.2.1 that is, as example, the algorithm for a device type A.

This new version work with two files *authorizedDevices.csv* and *devices.csv* despite the single file and the tuple with the characteristics, such as seen in the Pilot version. But, this version only implement the work the devices A and B, thus, they will not be used.

Another difference between the versions is the development focus. While the Pilot version has proposed a general experience with the technologies, this version is focused in explore the different devices working separately. This is better at development level because open spaces for the project be constructed in an incremental way, isolating the tasks.

Because this insight, the project has started to be developed around the three components already quoted, and this is the first part. Thus, this version implement the complete system with 4 inputs and 4 outputs working as described in the Section 3.1, Chapter 3.The GPIOs that has been used as input/output also change trying to simulate better the ideal system.

This improvement is seeking a program able to be embedded on the hardware described in the Section 4.1. Thus, has been made another protoboard assembly to simulate a total, shows in Figure 4.17 and with the pins changes shows in Table 4.7.

| **Inputs** | | **Outputs** | |
|---|---|---|---|
| Twilight Sensor | GPIO19 | Spotlight (White) | GPIO23 |
| Camera 0 | GPIO27 | Pedestrian Board (Green) | GPIO26 |
| Camera 1 | GPIO18 | Flash Light (Red) | GPIO24 |
| Obstacle Sensor | GPIO20 | Crosswalk Board (Yellow) | GPIO25 |

Table 4.7: GPIO pins used such as Inputs/Outputs in the Structural version.

This assembly is used on the another two components of this incremental development approach and it has real photos shown in the Appendix D.2.

The inputs has changed to three wires connected on GND bus (for LOW value) which can alternate to 3.3V (for HIGH values) according the desirable inputs. The LEDs wiring has changed because, according the Hardware proposed in Section 4.1, the output relays

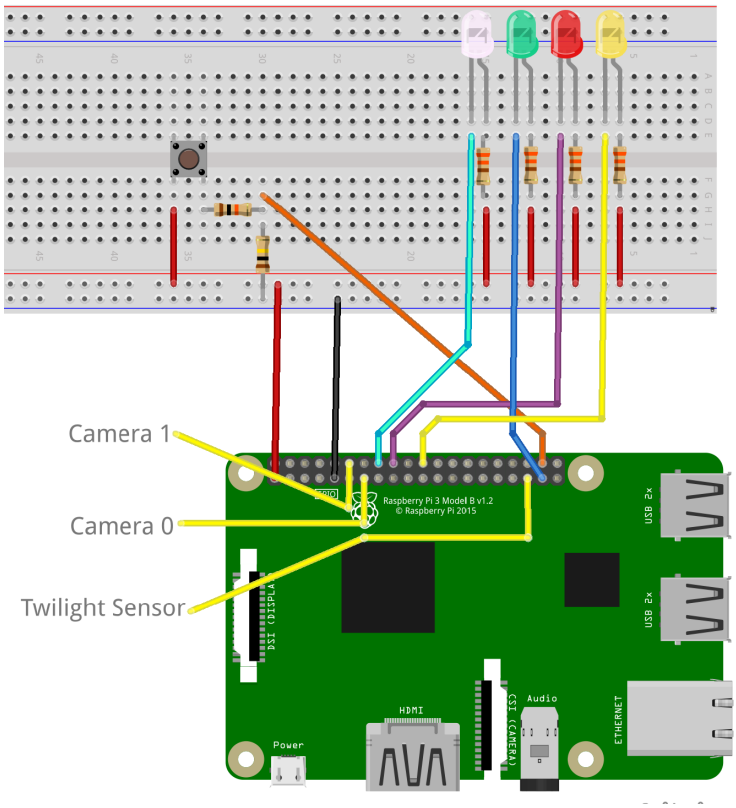



Figure 4.17: Structural, Communication and Integration assembly.

are activated with LOW values.

Because this, the LEDs polarity was inverted and, now, the anode is connected in the resistor wired on thd 3.3V bus and GPIO provides LOW values when a output must be enabled. Furthermore, the next topic will explain how the *RaspberryPi* class works besides explain their work inside the Main Routine.

### Program Work

Analyzing the *RaspberryPi* module line-to-line is possible to understand how it is created, which are their attributes and methods. The script responsible to create the class work as follow:

1. The algorithm call the necessary libraries (1-11).

2. Setup the class attributes such as the device's unique characteristics, the authorized

devices to establish connection, the GPIO pins which will be used as inputs/outputs (12-26).

3. Setup two status array, one to monitor the inputs' status of the own device and another to the inputs of the Pair Device (27-31).

4. All the Methods responsible to defines pins as input/outputs, inputs detection and activation/deactivation of outputs (28-110).

Also it is created helpful methods suach *_bounceTime* and *_allOutOff*, respectively, to implement delays and turn OFF all the outputs at the same time. Another details is, specifically, the Obstacle Avoidance Sensor because this is a input treated such as an interruption pin while the others are simple inputs.

This interruption pin detect the exactly moment when the pedestrian enter on the crosswalk and activate the Spotlight. In this way, save energy avoiding unnecessary activation, due their high consumption, such as seen in the Subsection 3.2.5. It helps in cases that the people just comes toward the crosswalk but does not enter on it.

For each device's type, the interruptions callback's conditions, on the line 84, verify if, at least one (Device A or B) Obstacle Sensor were activated. The code blocks stay in looping until that there are not more pedestrian crossing the street.

It also verifies if the Camera that covers the Area B" has detected any pedestrian on the crosswalk. But, if device is A type, the last condition depends of the Pair device status arrays. If it is B type, depends of their own status array.

Thus, after this brief explanation, an Object is created with this class framework and operate, divided in code blocks, such as follow:

1. Calls the necessary libraries (1-27).

2. Creates the *raspi* object, reading the .csv files and setup the inputs/outputs a defines the preset values (28-51).

3. It is the Main Routine which verify when and what output must be activated based on both status arrays already explained (29-76).

The Cameras are considered such as logical pins, but, in real circuits, their have specific libraries to work with. But this consideration is valid because, such as described in the paper seen on the Subsection 2.3.6, there are a lot of tools capable to detect pedestrians on crosswalks with local end-device's image processing.

In this way, this study sustain some hypothesis to employ image processing algorithms inside this project Raspberry Pi based. But, due the complexity increase, they are not used, but guarantee that this system can be applied in a real environment. Another detail are the main routine changes according to the device type.

For Device's type A, on the line 64, it verifies if there are cars on the Area A", according the Figure 3.7. It is responsible to pedestrian coming toward the crosswalk (Area A') and and cars waiting in front the crosswalk (Area A"). The program enter in this conditional code block based on their own status array, bearing in mind that just it has the camera which verifies the cars.

In contrast, for Devices' type B, the condition verify on the pair device's status array because. This device's type also detects pedestrian coming (Area B') but, despite the another type, it verifies if there are pedestrians on the Crosswalk (Area B"). This make it dependent of the Pair device status array to activate the Pedestrian Board.

### Challenges and Solutions

This version has created a Object class able to implement the smart device's work. The bigger challenge has been understand how this new programming approach should works. The library **raspi-lora** has been used as creation example for this Structural implementation.

In the first implementation of this version, the program did not work. The LEDs have been activated but they were blinking when the inputs were detect, not working well. In this way, has been created the Method *_bounceTime* which helps on the detection process. This prevents the floating values giving a soft delay when an input is verified.

So, after these considerations, the project has two different devices working separately but they need to connect themselves to implement all the system work environment. The

next implementation will develop isolated the RX continuous/TX to send explained on the Section 3.3.

### 4.2.4 Communication

The communication algorithm has been developed using the assembly seen in the Figure 4.15 to attach the RFM95W on the Raspberry Pi. It simulates the information changes between two devices in a RX continuous/TX to send method. The input/output is tested using the assembly seen in Figure 4.17.

#### Goals

Have been only considered the Twilight Sensor as input and the Spotlight as outputs seeking to simplify tests, following goals:

- Implement a RX continuous/TX to send method.
- Ensure a communication without faults.

This version is quicker when compared with the others it shown in the Appendix C.3.1. It is focused on the development of algorithm able to make the RFM95W receiving new packages all the time and only send messages when new input values on the status array are detected,

#### Approach

The approach is based in a inputs' detection flag. This Flag is True when a inputs is detected and False if it is not. Such as central logic pillar, it commands the chat between two devices dictating when a new update must be sent to the another device.

#### Program Work

The program use the library **raspi-lora** and work line-to-line, as follow:

- Calls the necessary libraries (1-8).
- Defines the GPIO that will be used, if they work as input or output and create *lora* (9-20).
- Setup all the variables that will be used (21-29).
- Start the Main Routine in constant looping (30-61).
  - Set mode RX constant and verify if a new package has been received and convert it for a Boolean value (34-44).
  - Verify if it has happens a status change based in the Twilight Sensor detection (45-51).
  - Still in RX mode if the Flag did not change, turn on/off the Spotlight according the status and, finally, send an command to the Pair device if the Twilight Sensor was detected (52-61).

**Challenges and Solutions**

The main challenge was the devices' synchrony. They lose it after communication tries. Another problem happens when the communication side changes, that is, who starts as sender and, after send, becomes receiver. With this changing, the outputs not answer in the right time.

This issue was solved adding the line 47, a delay on the Flag variation. This delay has solved the synchrony problem ensuring that will not be sent unnecessary status notifications which flood the communication channel.

Finally, his algorithm version is the second component of the incremental developing. The Integration Implementation will join the the Structural and the Communication versions seeking to make a base for the final algorithm version.

Despite the incremental developing has been validated version-to-version, when the Structural and the Communication version are combined new logic faults can happen.

The next Topic will discus the last considerations, the developed adaptations and it will gives a preview of the final version.

### 4.2.5 Integration

This is the final component of the incremental developing proposes. In contrast with the another Subsections explaining each goal, approach and challenges, this one will only shows how the other two components implement this final. In this way, it will focus how the integration works and why each change has been developed.

#### Program Work

The Integration version emulates the complete scenery which two Smart Crosswalks change information and both working only with the Twilight Sensor and the Crosswalk Board. The Twilight Sensor activates all the system and enable the Crosswalk Board when is night.

As well the Pilot implementation, this version works with a main script called *system.py* that interact with the modules that have been made in the Structural and the Communication implementations. They are organized as follow:

- **raspi**: Raspberry object class' creation module. It is compound by the **raspi.py** which is the script responsible for defines the object's attributes and methods and the *___init___.py* used for the module's work.

- **communication**: LoRa object class' creation module. It is compound by the *constant.py* script which is the module's constant, the **lora.py** which is the script responsible for defines the object's attributes and methods and, finally, the *___init___.py* used for the module's work.

- **tools**: Module with the tools to read and convert the data inside the files *device.csv* and *authorizedDevices.csv*.

The communication process uses the files *device.csv* and *authorizedDevices.csv* to setup their properties. This setup includes, respectively, the unique characteristics for each device, in the Table 4.8, and who is authorized to establish communication.

| Attributes | Unique IPv6 ID | Channel | Address | Type |
|---|---|---|---|---|
| Type A | c9ed:d191:4c2e:d978:644a:8c55:2fb4:27e | 0 | Sidarta-84 | A |
| Type B | 4649:6858:89ae:b771:e21e:f0b4:8a2e:e42a | 1 | Sidarta-84 | B |

Table 4.8: Devices' unique characteristics used on the Integration.

In this implementation the Unique IPv6 ID and the Channel Communication are used to make the communication protocol. This protocol, as already explained, first verifies if the attached ID in the package received exist in the authorized devices list. Besides this verification, all the messages also be encrypted, increasing the secure.

The attributes used in the class *lora* are setup such as the documentation of the library operating with cryptography and in the 868 MHz band, the unlicensed European Band. Bearing in mind this environment scenery the next topic will shows how the adaptations work and why they were made.

### Upgrade and Adaptations

The codes shown in Appendix C.4 are the system's Main Routine and the changes that were made in the *raspi.py*, proposed in the Appendix C.2, and in the *raspi-lora* library. First, the changes implemented on the *raspi.py*, looking all the lines in Appendix C.4.2, are such as follows:

- Attribute's creation called *_statusPair* and it is an array which contains all the inputs' status of the Pair device (Line 11).
- Attribute's creation *_flag* and it is responsible to detect when a new input is detected and the system must send a status update to the Pair device (Line 12).
- Method's creation called *_message* and it is responsible to code the message that

will be sent. This codification convert an status array from logic values (True or False) to an sequence with '1' or '0' (14-21).

- Method's creation called *_pairDetection* an it must makes the authorized devices' validation and update the *_statusPair* array (23-29).
- Method's creation called *_flagTest* an it should makes the *_flag* update based on a new input detection (31-39).

These attributes and methods compound the sending and receiving process in the *raspi* object's side. The adaptations made on the *raspi-lora* library (Appendix C.4.3) were to treat the received packages and preset attributes in the *lora* object's side as follow:

- All the values that must be given when the object is created are defined inside the class setup.
- Attribute's creation called *_thisID* which stores the Unique IPv6 ID (Line 6).
- Attribute's creation called *_this_address* which stores the communication channel (Line 16).
- Attribute's creation called *_message* and store the received message (Line 20).
- Cryptography activation using the key *the_suffering_is_the_way*. Both devices have the same key and it is used to encrypt and decrypt the packages (Line 21).
- Adaptation in the *on_recv* to make it decodes the received packages and update the the *_message*.

Therefore, *lora* object gives the received package to the *raspi* through the *_message* attribute. The *raspi* has a method to code the package and use a *lora* method, called *send*, to send it to the Pair device.

On the another hand, the *lora* has a method to decode the package and gives this information for a *raspi* method, called *_pairDetection*, to allow or not the package and store it in the *raspi* attribute called *_statusPair*.

The Main Routine (Appendix C.4.1) uses this communication process to test a Final version preview. It implements the communication flag based while activate the Crosswalk Board according the night detection. This test has resulted in a system working without faults for only one input activating one output. This routine are not explained because only integer the another two, Structural and Communication.

Thus, this last version finish the incremental developing and has check the easier case scenery. Nevertheless, it provides a base algorithm to implement the Final Embedded System that will be discussed in the Chapter 5. The next Subsection explain how to embed a system inside a Raspberry Pi in a way that it works autonomous in the outdoor environment.

### 4.2.6 Software Incorporation

The system needs to be embedded on the board to work in autonomous way. To achieve it, has been used the *cron*, a Linux's default software utility that is a time-based job scheduler [73].

But, despite this is the main tool, to install all the necessary features it is desirable to has basic knowledge in Linux Terminal commands to install packages and setup the cron. Thus, all the installation occurs using two devices, A and B, where includes the follow steps:

1. Connect the Raspberry Pi on the internet

2. Download the Smart Crosswalk's Setup Script C.6 and the both Systems Modules (A and B) C.5.

3. Run the Smart Crosswalk's Setup Script and generates the 'authorizedDevices_X.csv' and the 'device_X.csv' for both devices, where X = A or B

4. Cut the **System Module**, **'authorizedDevices_X.csv'** and **'device_X.csv'** for each devices type and paste these files on the directory **'/home/pi/smartCrosswalk/'** for both of them

5. Use the Raspbian terminal with the following commands:

   (a) **sudo su** and enter with it Raspberry credentials defined on Operating System installation

   (b) **curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py** to ensure that the *pip* package is installed

   (c) **apt-get install python3.6** to install the Python interpreter if it is not installed

   (d) **pip install cryptodome** to install the message encrypter package

   (e) **crontab -e** to call the scheduler software

6. In the *cron* work space write the command **@reboot cd /home/pi/smartCrosswalk/ && python3 /home/pi/smartCrosswalk/system.py**, save and return to terminal

7. Execute **raspi-config** to change the boot configurations

8. Follow the path "Boot Options → Desktop / CLI → Console Autologin" and when to close the raspi-config select <Yes> to reboot the microcontroller. This step setup the Raspberry Pi to execute the 'system.py' script always when this device reboot automatically

This step-by-step install the necessary python packages for the Embedded System works well. The Raspberry Pi connected on the internet is used to ensure, first, that the Python script is able to be embedded on the board.

Now, when the system restart, the Raspberry Pi will automatically executes the Smart Crosswalk System and will be able to operate on the outdoor environment. Despite this, it is necessary to has all the peripherals (inputs and outputs) attached on the Control Board, and, lastly after the power supply be wired to prevent algorithms errors.

# Chapter 5

# Results and Discussion

The next sections will be dedicated to the presentation of results regarding the Circuit Schematic and the Embedded System. The Circuit Schematic analyses will discuss how this circuit proposal can be useful, at which scenery it could be applied and about their energy autonomy cited in the Chapter 3.

The Embedded System analyses will describe the final algorithm implementation and the improvements made starting from the Integration version (Subsection C.4). Finally, will be simulated possible faults using the inputs which represent pedestrians' behavior to cross the street on crosswalks.

## 5.1 Hardware

The Circuit Schematic proposal in Appendix B, in fact, is a the physical part of a Smart Device to integer a lighting's Smart City environment. This device, within the IoT infrastructure concepts (Subsection 2.1.1), encompass the perception and the network layer, being capable to detect the environment around the model of crosswalk seen in Figure 3.2.

The final system is not able to communicate with the Middleware layer because it has been not implemented a communication with, for example, a gateway. Although the Embedded System not perform this communication, in the hardware level, this device has

the RFM95W integrated which provides structure enough for the next steps to develop a IoT service.

Furthermore, this possible implementation could be an IoT system within a SSL environment destined to crosswalks. The present result, within SL working structure (Subsection 2.2.1), only propose a LU which can be enabled (with embedded system's changes) to communicate with a LCU and, with the current peripherals, generate different reports.

Deepening on the idea of the cameras' use such as inputs, it is possible to develop a system which can use image processing tools to identify people and generates this reports, such as seen in Subsection 2.3.5 and 2.3.6.

Traffic accidents, pedestrians' flow intensity and traffic rules' violation are only some examples of data that could be acquitted and sold, in a report shape, to traffic mapping softwares, besides to get improvements in the Smart City's infrastructure, transportation and services pillars.

Another interesting spots are the system's energy autonomy and their input/outputs peripherals use. The ideal system consider the photovoltaic panels and the power consumption described in the Subsection 3.2.5 to evaluate, grossly, if is possible to implement an energetically autonomous system.

In order to do so, using the Equation 3.2 a the proposed panel values:

$$
\begin{aligned}
&= D\epsilon P_{jan} \\
&= (1.67)\left[m^2\right] \times (0.18) \times (53.1)\left[\frac{KWh}{m^2}\right]
\end{aligned}
$$

$$\therefore\ P = 15.93\ [KWh]$$

Therefore, $P$ is the amount of energy generated for the solar panels which restrict the system consumption. Applying the consumption values seen in Table 3.5 in the Equation 3.3 is possible to calculate the energy demands as below:

$$= \sum P_{cb} + \sum P_{in} + \sum P_{out}$$
$$= 2902 + 976.5 + 840$$

$$\therefore \; P_{con} \approx 4.8 \; [KWh]$$

Thus comparing the generated energy in the worst scenery and the system's power demand, the present work can be energetically autonomous. Furthermore, due the $P > P_{con}$ (almost $4\times$ lower), grossly, the project can use clean energy such as power supply source besides to work independent of the energy power from line grid. This evaluation only confirms if this crosswalk system can be deployed on the real environment, however, that is a superficial analyse due it is not one of this thesis goals.

## 5.2 Embedded System

This Final Version of the Embedded System, as explained in the Subsection 4.2.5, is an Integration version's improvement and has a pretty similar working when compared with this one.

### 5.2.1 Software

It follows the same goals, approach and program working, only suffering changes to overcome the problems found during the test that will be described in this Section.

#### Goals

- Implement an Embedded System applied in SSL directed to crosswalks.
- Develops a system should detect 4 inputs (Subsection 3.2.3) and reacts with 4 lighting outputs (Subsection 3.2.4) seeking ensure the pedestrian secure during the crossing process.

- A software prepared to be incorporated in the circuit schematic seen in Subsection 4.1, following the process seen in the Section 4.2.5.
- Develops a Embedded System with easy interpretation for future improvements.

**Approach**

- Two similar programs which must be integrated in two Smart Devices, one in each side of the crosswalk.
- Each device has unique characteristics such as a Unique IPv6 ID, a communication channel (integer number), an address and their type.
- The devices establish communication between themselves via LoRa sending encrypted packages with an inputs' status array and an authorized IPv6 ID.
- All the communication process is only allowed if the IPv6 ID attached on the message exist is registered in a list of authorized devices.
- The inputs evaluation to enable the outputs are lead by the device inputs' status array and their pair status array sent by message.
- The system only works during the night period.
- The inputs are:
  - Camera A' or B': Both of them are able to identify the area in front their respective sides of the crosswalk (Figure 3.2).
  - Camera A" or B": The first camera covers the are which verify if exist cars in movement and the second on ensure if there are pedestrians crossing the crosswalk (Figure 3.2).
  - Obstacle Avoidance Sensor: Detects the exact pedestrians' entrance moment on the crosswalk.
  - Twilight Sensor: Detects the night period.

- The outputs are:
  - Spotlight: Illuminates the crosswalk.
  - Flash Light: Alerts the driver when a pedestrian is near of the crosswalk (Area A' and B').
  - Pedestrian Board: Alerts the driver that exist a crosswalk near.
  - Crosswalk Board: Signalizes for the pedestrian when there is not cars coming and the right moment to enter on the crosswalk.

**Program Work**

This final implementation works with the 4 inputs/outputs bearing in mind the final approach and goals to develop the scripts in Appendix C.5. The assembly made to simulate and test this version is shown in Figure D.3. The pins used as inputs/outputs are the same which those applied on the Integration version (Appendix D.2).

The difference between the actual version and this base version is the homemade RFM95W breakout helpful to reduce interference's noises. Beside this, during the final developing it has been realized that the Obstacle Avoidance Sensor is a redundancy of the Camera responsible to detect the crosswalk (Area B").

Both perform the detection of the pedestrian entering on the crosswalk, thus, due this redundancy, this obstacle sensor was removed. This sensor can be helpful to deploy a less expensive version, because this, all the Embedded System and Circuit Schematic still supporting this input's peripheral.

The Main Algorithm has their implementation developed with variables, methods and attributes named to seek the easier interpretation for another developers. Finally, the programs is shows in the Appendix C.5.1, line-to-line, as bellow:

- Delay time to ensure the correct *cron* start when the system is enabled (Line 5).
- Imports the *raspi*, *communication* and *tools* modules (1-10).

- Creation of the objects *raspi* and *lora* (11-17).
- Set GPIO pins as inputs and outputs (18-27).
- Deactivates and ensure that the device will wait their pair to start the detection process (24-34).
- Main Routine which stay receiving messages continuously and send an update to the pair device always that a new input is detected. In sequence, it works as follows:
  - Detects new inputs and verify if the pair device has sent any inputs' status array (39-43).
  - Verify the flag which defines if any inputs has been detected, If none inputs have been detected the system enter in the RX continuous mode and stay verifying the inputs (39-83).
  - Send a message to the pair device with the current inputs' status array (84-86).

This algorithm follows the detection process explained in the Section 3.3.3, Figure 3.12 and it has been removed the Obstacle Sensor Avoidance before explained. In the end, each device type has an *system.py* script pretty similar and equal modules. Looking the lines block 62-64 there is the condition that make them different.

The program A, used as example, consider the status array from the pair of the device to activate the Spotlight. The program B consider their own inputs' status array. These singularities are due the System A be responsible to detect cars in the Area A" while the System B detects if there are pedestrians on the Area B", that is, on the crosswalk.

The Spotlight only is disabled when there are not more pedestrians crossing the street and, in this way, only the type B devices can order to the type A device turns off all the crosswalk lighting. Therefore, due this small difference it is only attached the type A device's code. To transform the Type A script in Type B, it just necessary to change, on the Line 62, the command raspi._statusPair for raspi._status.

### Improvements

The Appendix C.5 also shows the improvement made in the *raspi.py* as well within the *lora.py* module. In this way, are made the change in the Raspberry Pi modules (Appendix C.5.2) as follow:

- Has been created a file called *constants.py* with the inputs GPIO pin numbers, the indexes for the status array and the message. This improvement seek become the program more editable for another developers in future improvements.
- The constants were replace in their due methods and conditions codes in the main code.
- Method's creation called *_inputsDetection* to perform a generic function able to read the inputs. It also is capable to activate the flag which decides if the Smart Device must send or not a new inputs' status array (76-89).
- Method's creation called *_outputCommand* to perform a generic function able do activate/deactivate the outputs (91-94).

The *constants.py* script has transformed all the program in a easy editable script. The generic functions have reduced the number of code lines and they work with two variables. These values are an output pins' list instead of one different method for each peripherals (input/output), as has been made in the integration version.

The Appendix C.5.3 shows a method created in the *lora.py* to prevents out of synchrony communication. This function obligates the system to wait their pair device activation when connected on the power supply during the installation moment.

The default method *lora.send_to_wait()* (from the **raspi-lora** library) returns False when the pair of the device does not receive a message and True in the opposite case. This returned values implement the while logic. In this way, always that a device receive a message it sends a *acks* message.

This 'acknowledgment' message says if the sent package has been successful received. Using this functionality, the device constantly send and wait an empty message to start

to work. Dues the 'wait and send' process, the device opens a one second gap (in RX mode) for the it receives their empty message from their pair.

Finally, to finish this Final Algorithm description, will be shown in the next Topic the final challenges and solutions found during the developing process.

#### Challenges and Solutions

Some challenges that have been found after the Integration version's development are solved in this Final Version. They were, basically, delay problems between the pair of devices, bad connection and the unnecessary use of the Obstacle Avoidance Sensor.

The key ideas used to solve these issues already have been described in the Program Work's Topic. In the end, this Embedded System has shown satisfactory results and they will be detailed and discussed in the Subsection 5.2.2.

### 5.2.2 Tests

The tests have been made within hypothetical work scenarios which verifies thing such as the outputs' activation, communication protocol, synchrony and the proposal of crossing process (Figure 3.2).

All of these tests have been made using the Final Assembly (Appendix D.7) and wiring the GPIO inputs' pins on the 3.3V bus when the inputs are HIGH and on the GND bus when they are LOW.

In general, the tests were made with sequential activation of outputs according their respective activation conditions shown, in sequence, in the Main Routine (Appendix C.5.1) such as described in the System's Work Timeline (Figure 3.12).

The detection areas and peripherals are disposal according explained in the Section 3.2 and must happens synchronously between the pair of devices. In sequence, the detection successful process occurs when:

1. Detects the Twilight and start the system's work for both devices.

2. Activates the Crosswalk Board.

3. Wait for pedestrians show interest in cross the street going toward the crosswalk on the Areas A' or B'.

4. Activates the Flash in the Device A to alert the drivers when is detected people on the Areas A' or B'.

5. Allows the pedestrians' crossing if there are not movements on the Area A". That is, the image must be almost static, because, if not happens movements the cars are stopped or there are not cars.

6. Activates the Pedestrian Board.

7. Detects pedestrians on the Are B" and ensure the Spotlight enabled while there are pedestrians on the crosswalk.

8. Turns off the outputs according if detection not happens anymore.

Thus, following this step-by-step, has been verified that the Embedded System work well in the following cases:

- Despite the device already has been turned on, its only starts to detect when their pair also plugged on the power supply.
- Outputs' sequential activation according the previous explained sequence.
- Always when any inputs are activated the *raspi._flag* is updated.
- If the received message has an ID not registered in the *authorizedDevices.csv*, the program ignore the *raspi._statusPair* update.
- Send encrypted messages.
- Crossing process from the Side A to B and the reciprocal process (B to A).
- Hypothetical pedestrians which do not wait the cars stop on the Area A".

The wait for the pair device ensure that will not happens updates in the inputs' status array without their pair be notified. The input detection flag ensure that each device always knows when any of the six inputs has been activated. The program only accept messages from registered devices, beside this, it sends encrypted messages. These both security cares keep the system safe from malicious activities. In the last instance, the system ensure lighting for all cases, even in pedestrians' premature actions.

This thesis project offers a Embedded System compatible with the Circuit Schematic seen in Section 4.1. The system is strongly based in the fact that the cameras can detect pedestrians and cars presence. To implement a real system validation, would be necessary integrate this program with image processing algorithms and ensure the inputs detection process.

Despite have been researched studies which show that is possible to make it with local processing (Subsection 2.3.6, if happens faults in this detection approach, the pedestrians safety is compromised. Despite this, the program can work with alternatives, such as the Obstacle Sensor Avoidance, besides being easily editable.

The program has not been tested in practical environment, but, it proposes a autonomous system's work which can be deployed, moreover, within an IoT service. This final proposal works with six inputs and seven outputs, summing both devices' peripherals. For both devices, it has been removed the Obstacle Avoidance Sensor and the Type B device does not has the Flash Light.

Within these logical inputs' tests the system has a robust answering and it proposes the core logical function of an IoT service on a Smart City environment. It could generates data for a Middleware Layer which could integrate a helpful system capable to monitor the pedestrians and confirm their safety on the traffic.

# Chapter 6

# Conclusions and Future Works

## 6.1 Conclusion

Bad lighting conditions are one of the main causes which implies on traffic fatalities involving pedestrians. The present work has proposed a core logical function of a SSL infrastructure applied in crosswalks besides provide bases for a Smart Device's hardware development. The function script is programmed in Python and it is developed intending to ease future improvements. The tested cases achieve a synchronized response between the pair of devices and encompass even when the pedestrian do not follow the proposed crossing process. Furthermore, this performance has their hardware restrictions matching with the developed circuit schematic, which makes the system deployment feasible in practical environment. Also, the project is designed to fit in an IoT service and it can implement an useful solution, in different subjects, within Smart City environments. For example, this system can integrate a service which send data to a Control Unit which monitors accidents, people flow and traffic infractions. This data can be stored in a database indexed for each pair of devices localization and it generates statistics for flow, accidents and infractions intensity in different geographical places. With the correct Business Layer implementation, it could help in governmental programs to prevent these critical traffic issues. Finally, all this technological and smart proposal can help in the

local Portuguese scenery of traffic accidents providing a good crosswalk signalization as well a safe and illuminated path for elderly pedestrians.

## 6.2 Future Works

This work sustain ideas which suggest the follows future works:

- Case studies for pedestrians detection on crosswalks using cameras.
- Improvements in the algorithm work regarding automation to install it, sensible cases of inputs' detection and accessibility using new output approaches.
- Development and budget of a PCB project applied to smart lighting and signalization of crosswalks as well their viability to be deployed in large scale.
- Case studies for hardware restrictions such as communication noises and input/outputs peripherals limitations.
- Implement an IoT Service with autonomous architecture and able to generate reports regard traffic accidents, people flow and traffic infractions.
- Deploy this project idea on real world, bearing in mind all outputs, inputs and environmental issues.

# Bibliography


[1] European Commission, "Traffic Safety Basic Facts on Pedestrians", Tech. Rep., 2018, pp. 1–24. [Online]. Available: http://bit.ly/2NVBIa6 (visited on 01/27/2020).

[2] INE, "Estimativas de População Residente em Portugal", Tech. Rep., 2019, pp. 1–14. [Online]. Available: http://bit.ly/30SxEN9 (visited on 01/27/2020).

[3] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future Internet : The Internet of Things Architecture , Possible Applications and Key Challenges", 2012. DOI: 10.1109/FIT.2012.53.

[4] L. Tan and N. Wang, "Future Internet: The Internet of Things", *ICACTE 2010 - 2010 3rd International Conference on Advanced Computer Theory and Engineering, Proceedings*, vol. 5, pp. 376–380, 2010. DOI: 10.1109/ICACTE.2010.5579543.

[5] G. M. Lee, "The Internet of Things – A Problem Statement", pp. 517–518, 2010.

[6] S. Nuratch, "A universal microcontroller circuit and firmware design and implementation for iot-based real-Time measurement and control applications", *2017 International Electrical Engineering Congress, iEECON 2017*, 2017. DOI: 10.1109/IEECON.2017.8075906.

[7] "An information framework for creating a smart city through internet of things", *IEEE Internet of Things Journal*, vol. 1, no. 2, pp. 112–121, 2014, ISSN: 23274662.

[8] J. Robert, *Urban Europe*. 2018, pp. 3–69, ISBN: 9789279601392. DOI: 10.4324/9781315777962-2.

[9] European Commission, *Worldwide Urban Population Growth*. [Online]. Available: http://bit.ly/2OB1lgG (visited on 02/07/2020).

[10] S. Koutra and V. Becue, "A Multiscalar Approach for ' Smart City ' Planning", *2018 IEEE International Smart Cities Conference (ISC2)*, no. 1, pp. 1–7, 2018.

[11] L. G. Anthopoulos, *The Rise of the Smart City*, April 2017. 2018, ISBN: 9783319570150. DOI: 10.1007/978-3-319-57015-0.

[12] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges", *IEEE Communications Surveys and Tutorials*, vol. 18, no. 1, pp. 236–262, 2016. DOI: 10.1109/COMST.2015.2477041. eprint: 1509.07675.

[13] E. De Poorter, I. Moerman, and P. Demeester, "Enabling direct connectivity between heterogeneous objects in the internet of things through a network-service-oriented architecture", *Eurasip Journal on Wireless Communications and Networking*, vol. 2011, no. 1, pp. 1–14, 2011, ISSN: 16871499. DOI: 10.1186/1687-1499-2011-61.

[14] M. Castro, A. J. Jara, and A. F. Skarmeta, "Smart lighting solutions for smart cities", *Proceedings - 27th International Conference on Advanced Information Networking and Applications Workshops, WAINA 2013*, pp. 1374–1379, 2013. DOI: 10.1109/WAINA.2013.254.

[15] "IoT-enabled smart lighting systems for smart cities", *2018 IEEE 8th Annual Computing and Communication Workshop and Conference, CCWC 2018*, vol. 2018-January, pp. 639–645, 2018.

[16] R. B. García, G. V. Angulo, J. R. González, E. F. Tavizón, and J. I. H. Cardozo, "LED street lighting as a strategy for climate change mitigation at local government level", *Proceedings of the 4th IEEE Global Humanitarian Technology Conference, GHTC 2014*, pp. 345–349, 2014. DOI: 10.1109/GHTC.2014.6970303.

[17] G. T. Aditya and N. K. Prakash, "Smart and Efficient Outdoor Lighting System", *Proceedings of the 2nd International Conference on Intelligent Computing and Control Systems, ICICCS 2018*, no. Iciccs, pp. 1598–1602, 2019. DOI: 10.1109/ICCONS.2018.8663140.

[18] "An energy efficient pedestrian aware Smart Street Lighting system", *International Journal of Pervasive Computing and Communications*, vol. 7, no. 2, pp. 147–161, 2011, ISSN: 17427371. DOI: 10.1108/17427371111146437.

[19] W. F. Wang, "Study on several promising short-range wireless communication technologies", *2008 IEEE International Symposium on Knowledge Acquisition and Modeling Workshop Proceedings, KAM 2008*, pp. 727–730, 2008. DOI: 10.1109/KAMW.2008.4810593.

[20] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, "A study of Lora: Long range & low power networks for the internet of things", *Sensors (Switzerland)*, vol. 16, no. 9, pp. 1–18, 2016, ISSN: 14248220. DOI: 10.3390/s16091466.

[21] Z. Alliance, *Zigbee*. [Online]. Available: http://bit.ly/2IN2G0X (visited on 03/15/2020).

[22] L. R. Prando, E. R. De Lima, L. S. De Moraes, M. Biehl Hamerschmidt, and G. Fraindenraich, "Experimental performance comparison of emerging low power wide area networking (lpwan) technologies for iot", *IEEE 5th World Forum on Internet of Things, WF-IoT 2019 - Conference Proceedings*, pp. 905–908, 2019. DOI: 10.1109/WF-IoT.2019.8767343.

[23] A. Zanella and M. Zorzi, "Long-Range Communications in Unlicensed Bands: The Rising Stars in the IoT and Smart City Scenarios", no. October, pp. 60–67, 2016.

[24] Semtech, *LoRa*. [Online]. Available: http://bit.ly/3d1LAdu (visited on 03/15/2020).

[25] W.-S. Alliance, *Wi SUN*. [Online]. Available: http://bit.ly/38SFN6T (visited on 03/15/2020).

[26] Sigfox, *SIGFOX*. [Online]. Available: http://bit.ly/2TPMNNw (visited on 03/15/2020).

[27] Ingenu, *RPMA*. [Online]. Available: http://bit.ly/2QhyoHv (visited on 03/15/2020).

[28] Weightless, *Weightless*. [Online]. Available: http://bit.ly/2IMV9z3 (visited on 03/15/2020).

[29] I. ONE, *Dash 7*. [Online]. Available: http://bit.ly/2x1iwSF (visited on 03/15/2020).

[30] D. K. Srivatsa, B. Preethi, R. Parinitha, G. Sumana, and A. Kumar, "Smart street lights", *Proceedings - 2013 Texas Instruments India Educators' Conference, TIIEC 2013*, pp. 103–106, 2013. DOI: 10.1109/TIIEC.2013.25.

[31] T. Instruments, *MSP430x5xx and MSP430x6xx Family - User's Guide*. [Online]. Available: http://bit.ly/3b9r3lm (visited on 03/19/2020).

[32] N. P. Kumar and R. K. Jatoth, "Development of cloud based light intensity monitoring system using raspberry Pi", *2015 International Conference on Industrial Instrumentation and Control, ICIC 2015*, no. Icic, pp. 1356–1361, 2015. DOI: 10.1109/IIC.2015.7150959.

[33] J. A. Jang, H. S. Lee, and B. S. Yoo, "A real-Time sensing system of elderly's crossing behavior at outdoor environments", *2016 International Conference on Information and Communication Technology Convergence, ICTC 2016*, pp. 1143–1145, 2016. DOI: 10.1109/ICTC.2016.7763390.

[34] A. Adriansyah, A. W. Dani, and G. I. Nugraha, "Automation control and monitoring of public street lighting system based on internet of things", *ICECOS 2017 - Proceeding of 2017 International Conference on Electrical Engineering and Computer Science: Sustaining the Cultural Heritage Toward the Smart Environment for Better Future*, pp. 231–236, 2017. DOI: 10.1109/ICECOS.2017.8167140.

[35] Ubidots, *Ubidots: IoT platform*. [Online]. Available: http://bit.ly/2Wql5IJ (visited on 03/19/2020).

[36] K. E. An, S. W. Lee, Y. J. Jeong, and D. Seo, "Pedestrian-safe smart crossing system based on IoT with object tracking", *Lecture Notes in Electrical Engineering*, vol. 474, pp. 926–931, 2018, ISSN: 18761119. DOI: 10.1007/978-981-10-7605-3_147.

[37] M. Saleh, A. Muthanna, and Y. T. Lyachek, "Smart System of a Real-Time Pedestrian Detection for Smart City", *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pp. 45–50, 2020.

[38] "Walking pedestrian recognition", *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 1, no. 3, pp. 292–297, 1999. DOI: 10.1109/itsc.1999.821069.

[39] A. Rosebrock, *Deep learning on the Raspberry Pi with OpenCV*. [Online]. Available: https://bit.ly/391qdG0 (visited on 03/21/2020).

[40] Accuware, *Sentinel Introduction*. [Online]. Available: https://bit.ly/2QyVlWY (visited on 03/21/2020).

[41] *Kerlink - Communication is Everything*. [Online]. Available: https://bit.ly/2J1bWhW (visited on 03/20/2020).

[42] *A2A Smart City*. [Online]. Available: https://bit.ly/3dhUt2K (visited on 03/20/2020).

[43] *KiCAD like a Pro 2*. [Online]. Available: https://bit.ly/2Wbx7Fo (visited on 05/06/2020).

[44] *About KiCAD*. [Online]. Available: https://bit.ly/2X5Gdo1 (visited on 04/03/2020).

[45] *About Instructables*. [Online]. Available: https://bit.ly/3dM816V (visited on 04/03/2020).

[46] *Design Your Own Raspberry Pi Compute Module PCB*. [Online]. Available: https://bit.ly/34go2NP (visited on 01/21/2020).

[47] *Compute Module IO Board*. [Online]. Available: https://bit.ly/3aY4WOr (visited on 05/03/2020).

[48] *Flashing the Compute Module eMMC*. [Online]. Available: https://bit.ly/2YvBQ6m (visited on 05/03/2020).

[49] *Raspberry Pi Compute Module 3+*. [Online]. Available: https://bit.ly/3dSBoEF (visited on 04/06/2020).

[50] *Raspberry Pi 3 Model B+*. [Online]. Available: https://bit.ly/2XonAvV (visited on 04/06/2020).

[51] *RFM95W Transceivers*. [Online]. Available: https://bit.ly/2yyv9FG (visited on 04/06/2020).

[52] *Serial Peripheral Interface*. [Online]. Available: https://bit.ly/35BUOd4 (visited on 05/06/2020).

[53] *Photovoltaic Geographical Information System*. [Online]. Available: https://bit.ly/3e0Jr21 (visited on 04/06/2020).

[54] *Crosswalk Example*. [Online]. Available: https://bit.ly/2RyeKYL (visited on 04/06/2020).

[55] *REC TwinPeak 2 Mono*. [Online]. Available: https://bit.ly/2JNeSiE (visited on 04/06/2020).

[56] *Top 18 Solar Panel Manufacturers in Europe*. [Online]. Available: https://bit.ly/2x9olxZ (visited on 04/06/2020).

[57] *Camera Module*. [Online]. Available: https://bit.ly/2xDk85T (visited on 05/05/2020).

[58] *Obstacle Avoidance Sensor*. [Online]. Available: https://bit.ly/2z9MhCa (visited on 05/05/2020).

[59] *Twilight Sensor Model*. [Online]. Available: https://bit.ly/3fmLUET (visited on 05/05/2020).

[60] *Light Boards Model*. [Online]. Available: https://bit.ly/2SV0h9Z (visited on 05/05/2020).

[61] *Flash Light Model*. [Online]. Available: https://bit.ly/2WsrQbq (visited on 05/05/2020).

[62] *Spotlight Model*. [Online]. Available: https://bit.ly/2zdqCJj (visited on 05/05/2020).

[63] *Object-Oriented Programming in Python*. [Online]. Available: https://bit.ly/2W8Nliw (visited on 05/06/2020).

[64] *Raspbian*. [Online]. Available: `https://bit.ly/35BY5sM` (visited on 05/06/2020).

[65] *Python*. [Online]. Available: `https://bit.ly/2L8T5T2` (visited on 05/06/2020).

[66] *Angry IP Scanner*. [Online]. Available: `https://bit.ly/3ccJTsS` (visited on 05/06/2020).

[67] *Putty*. [Online]. Available: `https://bit.ly/2A7lM0D` (visited on 05/06/2020).

[68] *VNC Viewer*. [Online]. Available: `https://bit.ly/2L6Ns83` (visited on 05/06/2020).

[69] *Compute Module Hardware Design Guide*. [Online]. Available: `https://bit.ly/3fupa5Q` (visited on 05/09/2020).

[70] *Installing operating system images*. [Online]. Available: `https://bit.ly/2SSFwLV` (visited on 05/10/2020).

[71] *raspi-lora*. [Online]. Available: `https://bit.ly/3bF86Hl` (visited on 05/17/2020).

[72] *GPIOZero*. [Online]. Available: `https://bit.ly/3fYgxRa` (visited on 05/17/2020).

[73] *cron: Time-based Job Scheduler*. [Online]. Available: `https://bit.ly/3fKjDs3` (visited on 05/13/2020).

# Appendix A

# Raspberry Pi Compute Module Base Project

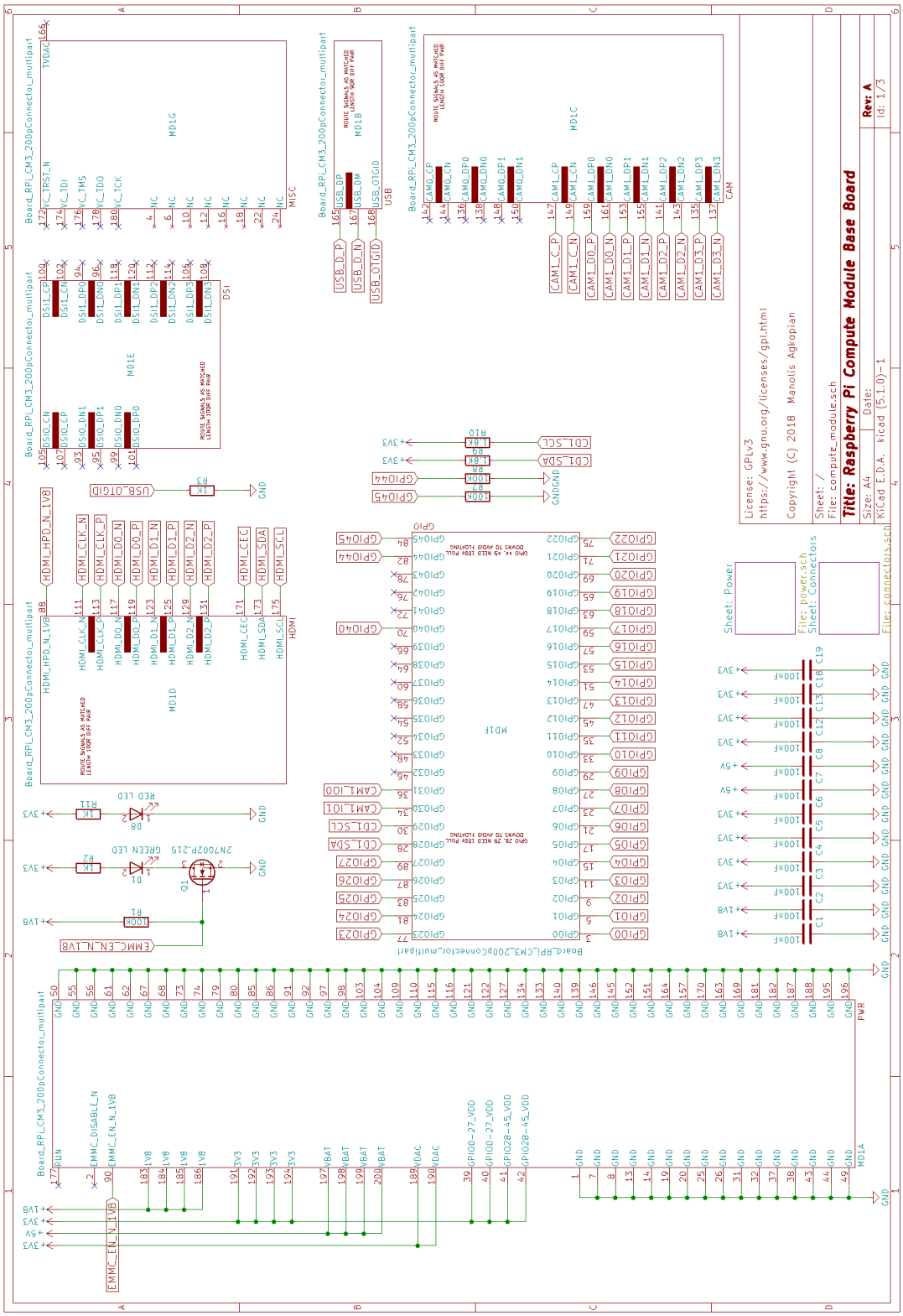



Figure A.1: Raspberry Pi CM3 of the original schematic.

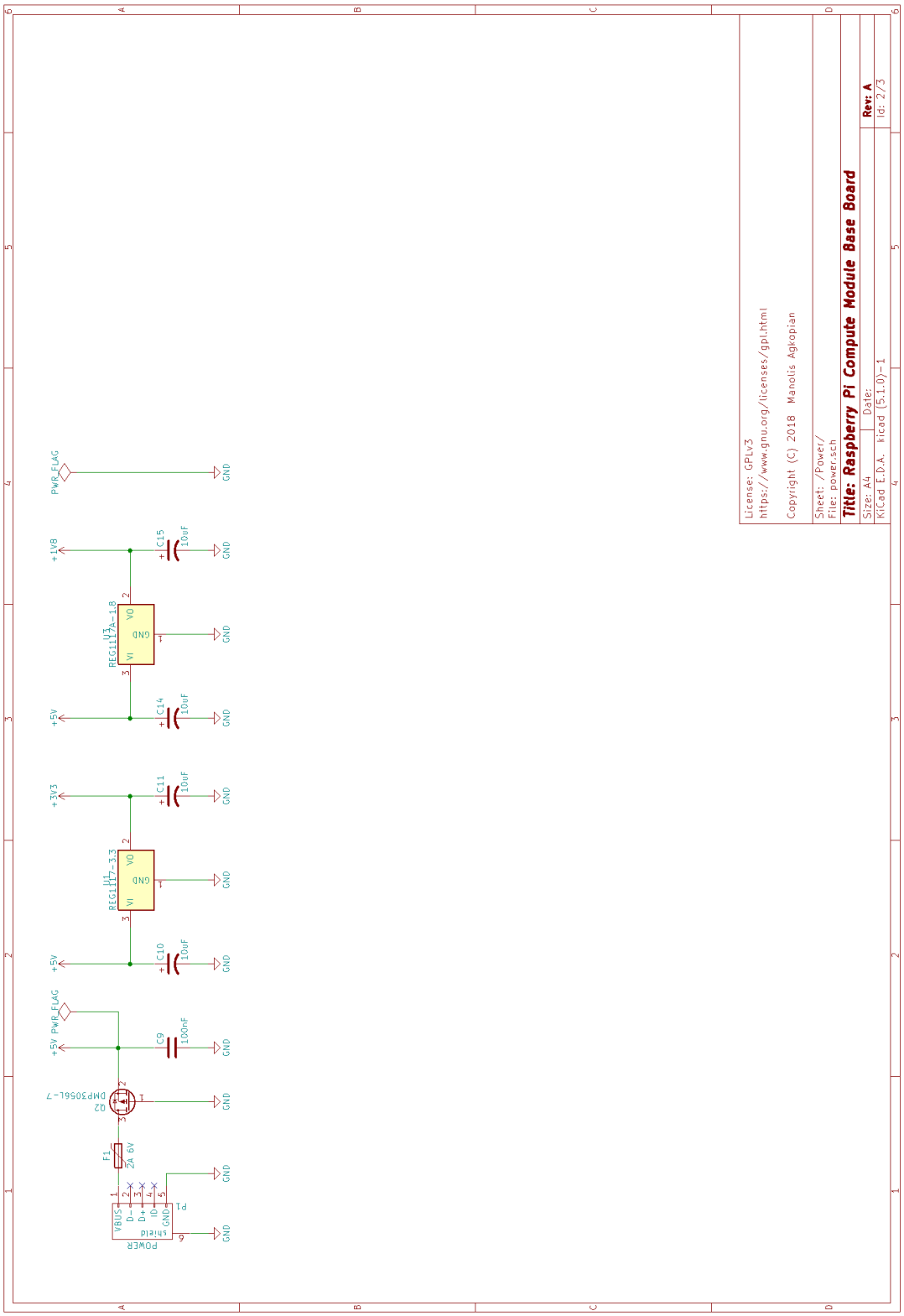



Figure A.2: Power regulators of the original schematic.

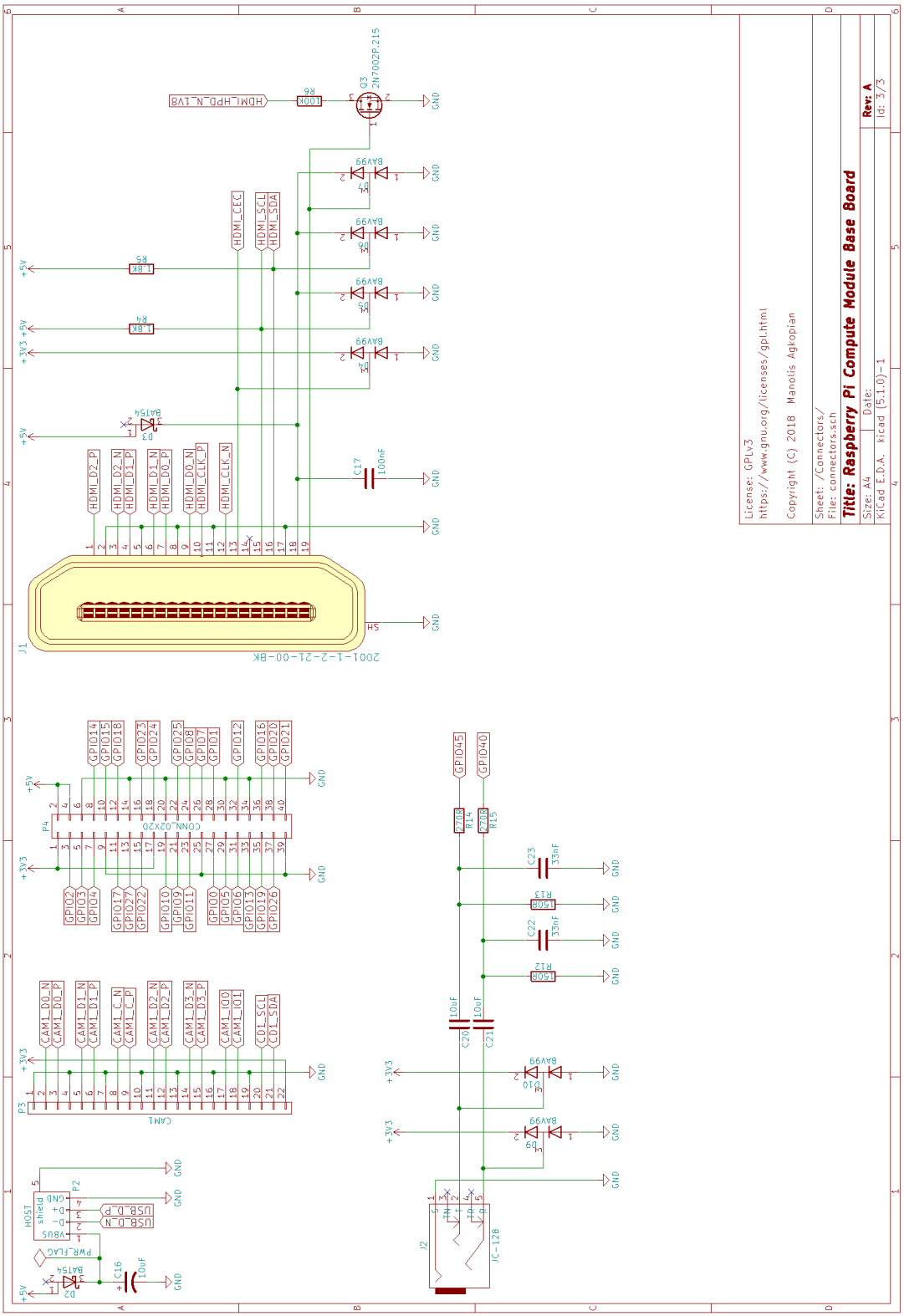



Figure A.3: Connectors of the original schematic.

# Appendix B

# Smart Crosswalk Schematic

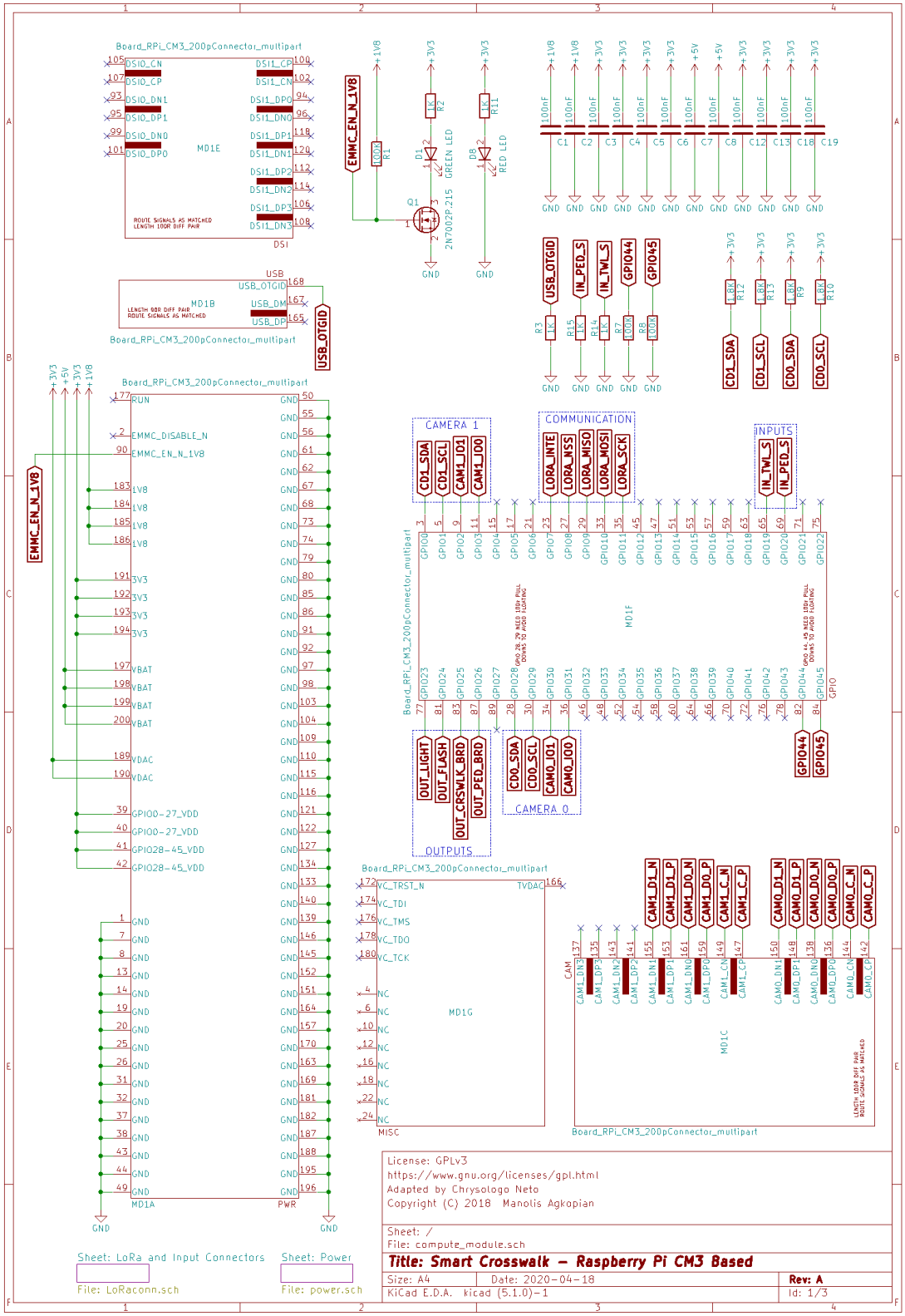



Figure B.1: Raspberry Pi CM3 of the developed schematic.

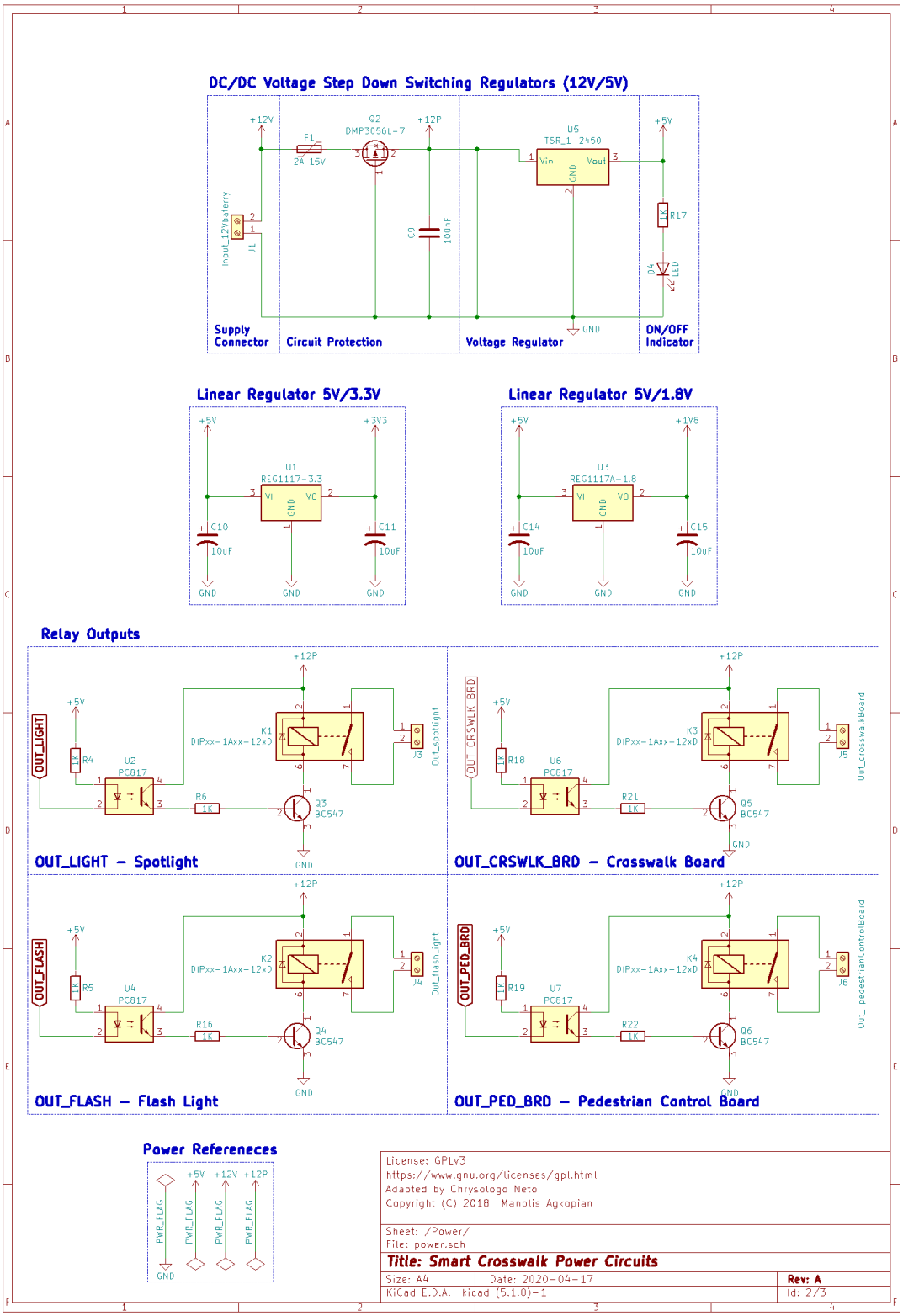



Figure B.2: Power regulators and outputs of the developed schematic.

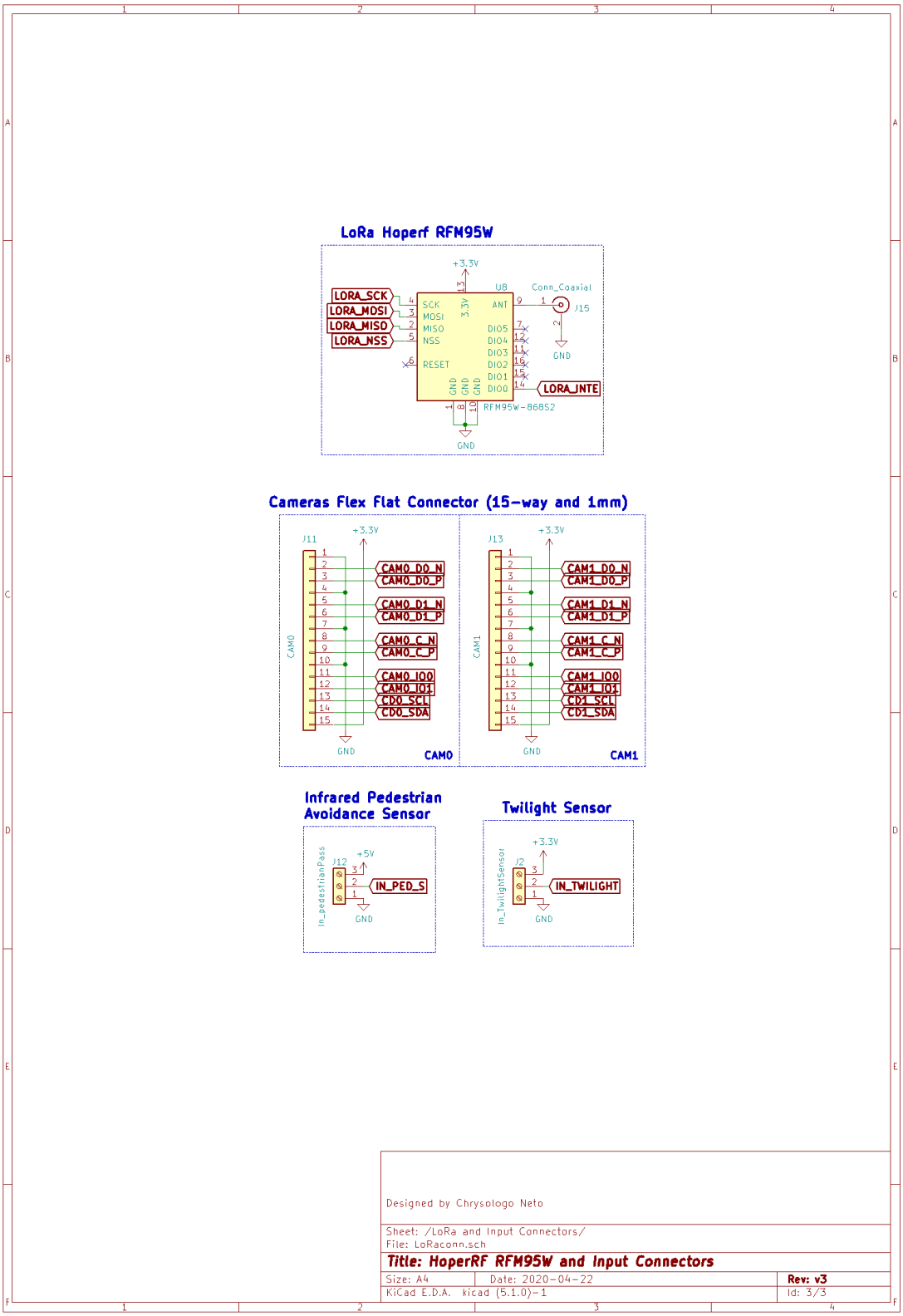



Figure B.3: LoRa RFM95W and connectors of the developed schematic.

# Appendix C

# Algorithm's Versioning

## C.1 Pilot

### C.1.1 Main Routine

```
#-------------------------------------------------------------#
## >>>>>>>>>>>>>          LIBRARIES         <<<<<<<<<<<<< ##
#-------------------------------------------------------------#

import RPi.GPIO as gpio
from lib.raspi_lora import LoRa, ModemConfig
from lib.functions import *
from lib.registers import *
import csv
import time

#-------------------------------------------------------------#
## >>>>>>>>>>>>>         PINS SETUP         <<<<<<<<<<<<< ##
#-------------------------------------------------------------#

[PIN_system, PIN_sensor, PIN_crosswalkBoard, PIN_spotlight,
   PIN_pedestrianBoard, PIN_flashLight] = [14, 0, 1, 2, 3, 4]

```

```
18 ## >>>>>>>>>>>>> DEFINE INPUTS/OUTPUTS <<<<<<<<<<<<< ##
19
20 pinMode(PIN_sensor, PIN_system, modo = gpio.IN)
21 pinMode(PIN_crosswalkBoard, PIN_spotlight, PIN_pedestrianBoard,
       PIN_flashLight, modo = gpio.OUT)
22
23 ## >>>>>>>>>>>> DEVICES CHARACTERISTICS <<<<<<<<<<<< ##
24
25 devicesID = read() # Authorized devices list
26 SMARTCROSSWALK = ('c9ed:d191:4c2e:d978:644a:8c55:2fb4:27e', 0,'SIDARTA
       AVENUE', 3)
27       # = (ID, channel, address, tempoCross)
28 lora = LoRa(0, 17, SMARTCROSSWALK[1], modem_config=ModemConfig.
       Bw125Cr45Sf128, tx_power=14, acks=True)
29 #    = LoRa(CE0 or CE1, interrupt, this_address, tx_power=14,
30 #           modem_config=ModemConfig.Bw125Cr45Sf128, acks=False, crypto=
       None)
31
32 ## >>>>>>>>>>>>>      PRESET VALUES     <<<<<<<<<<<<< ##
33
34 digitalWrite(PIN_crosswalkBoard, PIN_spotlight, PIN_pedestrianBoard,
       PIN_flashLight, nivel = gpio.LOW)
35 status = [None, None, None, None, None]
36 package = [None, [None,None]]
37 id_package = 0
38
39 #-------------------------------------------------------------#
40 ## >>>>>>>>>>>>>      MAIN ALGORITHM      <<<<<<<<<<<<< ##
41 #-------------------------------------------------------------#
42
43 while True:
44
45         gpio.wait_for_edge(PIN_system, gpio.RISING)
46         status[0] = True
47         crosswalkBoard(PIN_crosswalkBoard, status)
```

```
48          time.sleep(0.5)
49
50          while status[0]:
51
52              lora.set_mode_rx()
53
54              if lora._last_payload is not  None:
55
56                  receivedPackage = decPackage(lora._last_payload.message)
57
58                  for idRegistered in devicesID:
59                          if receivedPackage[0] == idRegistered[0] and
    receivedPackage[2] != package[1][0]:
60                                  package[0] = receivedPackage[1]
61                                  package[1][1] = package[1][0]
62                                  package[1][0] = receivedPackage[2]
63                                  break
64
65              if gpio.input(PIN_sensor) or package[0]:
66
67                  status[1:] = [True, True, True, True]
68
69                  if status[4] and (not package[0]):
70                      id_package += 1
71                      sensorPair = status[4]
72                      msg = codPackage(VALPASS[0], sensorPair, id_package)
73                      for channel in devicesID:
74                              lora.send(msg, channel[1])
75
76                  flashLight(PIN_flashLight, status)
77                  spotlight(PIN_spotlight, status)
78                  pedestrianBoard(PIN_pedestrianBoard, status,
    SMARTCROSSWALK[3])
79
80                  status[1:4] = [False, False, False, False]
```

```
                package[0] = False

                flashLight(PIN_flashLight, status)
                spotlight(PIN_spotlight, status)

#-------------------------------------------------------------#
## >>>>>>>>>>>>>   DEVICE'S DEACTIVATION  <<<<<<<<<<<<< ##
#-------------------------------------------------------------#

            if gpio.input(PIN_system):
                status[0] = False
                crosswalkBoard(PIN_crosswalkBoard, status)
                receivedPackage = [False, 0]
                time.sleep(0.5)
                break
```

### C.1.2 LoRa Module

**___init___.py**

```
from .lora import LoRa, ModemConfig
```

**lora.py**

```
import time
from enum import Enum
import math
from collections import namedtuple
from random import random

import RPi.GPIO as GPIO
import spidev

from .constants import *


```

```
13 class ModemConfig(Enum):
14     Bw125Cr45Sf128 = (0x72, 0x74, 0x04)
15     Bw500Cr45Sf128 = (0x92, 0x74, 0x04)
16     Bw31_25Cr48Sf512 = (0x48, 0x94, 0x04)
17     Bw125Cr48Sf4096 = (0x78, 0xc4, 0x0c)
18 
19 
20 class LoRa(object):
21     def __init__(self, channel, interrupt, this_address, freq=868,
    tx_power=14,
22                  modem_config=ModemConfig.Bw125Cr45Sf128, receive_all=
    False,
23                  acks=True, crypto=None):
24 
25         self._channel = channel
26         self._interrupt = interrupt
27 
28         self._mode = None
29         self._cad = None
30         self._freq = freq
31         self._tx_power = tx_power
32         self._modem_config = modem_config
33         self._receive_all = receive_all
34         self._acks = acks
35 
36         self._this_address = this_address
37         self._last_header_id = 0
38 
39         self._last_payload = None
40         self.crypto = crypto
41 
42         self.cad_timeout = 0
43         self.send_retries = 2
44         self.wait_packet_sent_timeout = 0.2
45         self.retry_timeout = 0.2
```

```
46
47          # Setup the module
48          GPIO.setmode(GPIO.BCM)
49          GPIO.setup(self._interrupt, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
50          GPIO.add_event_detect(self._interrupt, GPIO.RISING, callback=
    self._handle_interrupt)
51
52          self.spi = spidev.SpiDev()
53          self.spi.open(0, self._channel)
54          self.spi.max_speed_hz = 5000000
55
56          self._spi_write(REG_01_OP_MODE, MODE_SLEEP | LONG_RANGE_MODE)
57          time.sleep(0.1)
58
59          assert self._spi_read(REG_01_OP_MODE) == (MODE_SLEEP |
    LONG_RANGE_MODE), \
60              "LoRa initialization failed"
61
62          self._spi_write(REG_0E_FIFO_TX_BASE_ADDR, 0)
63          self._spi_write(REG_0F_FIFO_RX_BASE_ADDR, 0)
64
65          self.set_mode_idle()
66
67          # set modem config (Bw125Cr45Sf128)
68          self._spi_write(REG_1D_MODEM_CONFIG1, self._modem_config.value
    [0])
69          self._spi_write(REG_1E_MODEM_CONFIG2, self._modem_config.value
    [1])
70          self._spi_write(REG_26_MODEM_CONFIG3, self._modem_config.value
    [2])
71
72          # set preamble length (8)
73          self._spi_write(REG_20_PREAMBLE_MSB, 0)
74          self._spi_write(REG_21_PREAMBLE_LSB, 8)
75
```

```
76          # set frequency
77          frf = int((self._freq * 1000000.0) / FSTEP)
78          self._spi_write(REG_06_FRF_MSB, (frf >> 16) & 0xff)
79          self._spi_write(REG_07_FRF_MID, (frf >> 8) & 0xff)
80          self._spi_write(REG_08_FRF_LSB, frf & 0xff)
81
82          # Set tx power
83          if self._tx_power < 5:
84              self._tx_power = 5
85          if self._tx_power > 23:
86              self._tx_power = 23
87
88          if self._tx_power < 20:
89              self._spi_write(REG_4D_PA_DAC, PA_DAC_ENABLE)
90              self._tx_power -= 3
91          else:
92              self._spi_write(REG_4D_PA_DAC, PA_DAC_DISABLE)
93
94          self._spi_write(REG_09_PA_CONFIG, PA_SELECT | (self._tx_power -
    5))
95
96      def on_recv(self, payload):
97          print("Emissor: Canal ", payload.header_from)
98          print("Mensagem:", payload.message)
99          print("RSSI: {}; SNR: {}".format(payload.rssi, payload.snr))
100
101
102     def sleep(self):
103         if self._mode != MODE_SLEEP:
104             self._spi_write(REG_01_OP_MODE, MODE_SLEEP)
105             self._mode = MODE_SLEEP
106
107     def set_mode_tx(self):
108         if self._mode != MODE_TX:
109             self._spi_write(REG_01_OP_MODE, MODE_TX)
```

```
110             self._spi_write(REG_40_DIO_MAPPING1, 0x40)  # Interrupt on
    TxDone
111             self._mode = MODE_TX
112
113     def set_mode_rx(self):
114         if self._mode != MODE_RXCONTINUOUS:
115             self._spi_write(REG_01_OP_MODE, MODE_RXCONTINUOUS)
116             self._spi_write(REG_40_DIO_MAPPING1, 0x00)  # Interrupt on
    RxDone
117             self._mode = MODE_RXCONTINUOUS
118
119     def set_mode_cad(self):
120         if self._mode != MODE_CAD:
121             self._spi_write(REG_01_OP_MODE, MODE_CAD)
122             self._spi_write(REG_40_DIO_MAPPING1, 0x80)  # Interrupt on
    CadDone
123             self._mode = MODE_CAD
124
125     def _is_channel_active(self):
126         self.set_mode_cad()
127
128         while self._mode == MODE_CAD:
129             yield
130
131         return self._cad
132
133     def wait_cad(self):
134         if not self.cad_timeout:
135             return True
136
137         start = time.time()
138         for status in self._is_channel_active():
139             if time.time() - start < self.cad_timeout:
140                 return False
141
```

```
142             if status is None:
143                 time.sleep(0.1)
144                 continue
145             else:
146                 return status
147 
148     def wait_packet_sent(self):
149         # wait for `_handle_interrupt` to switch the mode back
150         start = time.time()
151         while time.time() - start < self.wait_packet_sent_timeout:
152             if self._mode != MODE_TX:
153                 return True
154 
155         return False
156 
157     def set_mode_idle(self):
158         if self._mode != MODE_STDBY:
159             self._spi_write(REG_01_OP_MODE, MODE_STDBY)
160             self._mode = MODE_STDBY
161 
162     def send(self, data, header_to, header_id=0, header_flags=0):
163         self.wait_packet_sent()
164         self.set_mode_idle()
165         self.wait_cad()
166 
167         header = [header_to, self._this_address, header_id, header_flags
    ]
168         if type(data) == int:
169             data = [data]
170         elif type(data) == bytes:
171             data = [p for p in data]
172         elif type(data) == str:
173             data = [ord(s) for s in data]
174 
175         if self.crypto:
```

```
176             data = [b for b in self._encrypt(bytes(data))]
177 
178         payload = header + data
179 
180         self._spi_write(REG_0D_FIFO_ADDR_PTR, 0)
181         self._spi_write(REG_00_FIFO, payload)
182         self._spi_write(REG_22_PAYLOAD_LENGTH, len(payload))
183 
184         self.set_mode_tx()
185         return True
186 
187     def send_to_wait(self, data, header_to, header_flags=0, retries=3):
188         self._last_header_id += 1
189 
190         for _ in range(retries + 1):
191             self.send(data, header_to, header_id=self._last_header_id,
    header_flags=header_flags)
192             self.set_mode_rx()
193 
194             if header_to == BROADCAST_ADDRESS:  # Don't wait for acks
    from a broadcast message
195                 return True
196 
197             start = time.time()
198             while time.time() - start < self.retry_timeout + (self.
    retry_timeout * random()):
199                 if self._last_payload:
200                     if self._last_payload.header_to == self.
    _this_address and \
201                             self._last_payload.header_flags & FLAGS_ACK
    and \
202                             self._last_payload.header_id == self.
    _last_header_id:
203 
204                         # We got an ACK
```

```
205                         return True
206         return False
207
208     def send_ack(self, header_to, header_id):
209         self.send(b'!', header_to, header_id, FLAGS_ACK)
210         self.wait_packet_sent()
211
212     def _spi_write(self, register, payload):
213         if type(payload) == int:
214             payload = [payload]
215         elif type(payload) == bytes:
216             payload = [p for p in payload]
217         elif type(payload) == str:
218             payload = [ord(s) for s in payload]
219
220         self.spi.xfer([register | 0x80] + payload)
221
222     def _spi_read(self, register, length=1):
223         if length == 1:
224             return self.spi.xfer([register] + [0] * length)[1]
225         else:
226             return self.spi.xfer([register] + [0] * length)[1:]
227
228     def _decrypt(self, message):
229         decrypted_msg = self.crypto.decrypt(message)
230         msg_length = decrypted_msg[0]
231         return decrypted_msg[1:msg_length + 1]
232
233     def _encrypt(self, message):
234         msg_length = len(message)
235         padding = bytes(((math.ceil((msg_length + 1) / 16) * 16) - (
    msg_length + 1)) * [0])
236         msg_bytes = bytes([msg_length]) + message + padding
237         encrypted_msg = self.crypto.encrypt(msg_bytes)
238         return encrypted_msg
```

```
239
240     def _handle_interrupt(self, channel):
241         irq_flags = self._spi_read(REG_12_IRQ_FLAGS)
242
243         if self._mode == MODE_RXCONTINUOUS and (irq_flags & RX_DONE):
244             packet_len = self._spi_read(REG_13_RX_NB_BYTES)
245             self._spi_write(REG_0D_FIFO_ADDR_PTR, self._spi_read(
    REG_10_FIFO_RX_CURRENT_ADDR))
246
247             packet = self._spi_read(REG_00_FIFO, packet_len)
248             self._spi_write(REG_12_IRQ_FLAGS, 0xff)  # Clear all IRQ
    flags
249
250             snr = self._spi_read(REG_19_PKT_SNR_VALUE) / 4
251             rssi = self._spi_read(REG_1A_PKT_RSSI_VALUE)
252
253             if snr < 0:
254                 rssi = snr + rssi
255             else:
256                 rssi = rssi * 16 / 15
257
258             if self._freq >= 779:
259                 rssi = round(rssi - 157, 2)
260             else:
261                 rssi = round(rssi - 164, 2)
262
263             if packet_len >= 4:
264                 header_to = packet[0]
265                 header_from = packet[1]
266                 header_id = packet[2]
267                 header_flags = packet[3]
268                 message = bytes(packet[4:]) if packet_len > 4 else b''
269
270                 if self._this_address != header_to or self._receive_all
    is True:
```

```
271                     return
272
273                 if self.crypto and len(message) % 16 == 0:
274                     message = self._decrypt(message)
275
276                 if self._acks and header_to == self._this_address and
    not header_flags & FLAGS_ACK:
277                     self.send_ack(header_from, header_id)
278
279                 self.set_mode_rx()
280
281                 self._last_payload = namedtuple(
282                     "Payload",
283                     ['message', 'header_to', 'header_from', 'header_id',
     'header_flags', 'rssi', 'snr']
284                 )(message, header_to, header_from, header_id,
    header_flags, rssi, snr)
285
286                 if not header_flags & FLAGS_ACK:
287                     self.on_recv(self._last_payload)
288
289         elif self._mode == MODE_TX and (irq_flags & TX_DONE):
290             self.set_mode_idle()
291
292         elif self._mode == MODE_CAD and (irq_flags & CAD_DONE):
293             self._cad = irq_flags & CAD_DETECTED
294             self.set_mode_idle()
295
296         self._spi_write(REG_12_IRQ_FLAGS, 0xff)
297
298     def close(self):
299         GPIO.cleanup()
300         self.spi.close()
```

**constants.py**

```
from enum import Enum


class ModemConfig(Enum):
    Bw125Cr45Sf128 = (0x72, 0x74, 0x04)
    Bw500Cr45Sf128 = (0x92, 0x74, 0x04)
    Bw31_25Cr48Sf512 = (0x48, 0x94, 0x04)
    Bw125Cr48Sf4096 = (0x78, 0xc4, 0x0c)


FLAGS_ACK = 0x80
BROADCAST_ADDRESS = 255

REG_00_FIFO = 0x00
REG_01_OP_MODE = 0x01
REG_06_FRF_MSB = 0x06
REG_07_FRF_MID = 0x07
REG_08_FRF_LSB = 0x08
REG_0E_FIFO_TX_BASE_ADDR = 0x0e
REG_0F_FIFO_RX_BASE_ADDR = 0x0f
REG_10_FIFO_RX_CURRENT_ADDR = 0x10
REG_12_IRQ_FLAGS = 0x12
REG_13_RX_NB_BYTES = 0x13
REG_1D_MODEM_CONFIG1 = 0x1d
REG_1E_MODEM_CONFIG2 = 0x1e
REG_19_PKT_SNR_VALUE = 0x19
REG_1A_PKT_RSSI_VALUE = 0x1a
REG_20_PREAMBLE_MSB = 0x20
REG_21_PREAMBLE_LSB = 0x21
REG_22_PAYLOAD_LENGTH = 0x22
REG_26_MODEM_CONFIG3 = 0x26

REG_4D_PA_DAC = 0x4d
REG_40_DIO_MAPPING1 = 0x40
REG_0D_FIFO_ADDR_PTR = 0x0d
```

```
36
37 PA_DAC_ENABLE = 0x07
38 PA_DAC_DISABLE = 0x04
39 PA_SELECT = 0x80
40
41 CAD_DETECTED_MASK = 0x01
42 RX_DONE = 0x40
43 TX_DONE = 0x08
44 CAD_DONE = 0x04
45 CAD_DETECTED = 0x01
46
47 LONG_RANGE_MODE = 0x80
48 MODE_SLEEP = 0x00
49 MODE_STDBY = 0x01
50 MODE_TX = 0x03
51 MODE_RXCONTINUOUS = 0x05
52 MODE_CAD = 0x07
53
54 REG_09_PA_CONFIG = 0x09
55 FXOSC = 32000000.0
56 FSTEP = (FXOSC / 524288)
```

### C.1.3 Functions

#### ___init___.py

```
1 import RPi.GPIO as gpio
2 from .confPins import *
3 from .codDecodPackages import *
4 import time
5
6 def crosswalkBoard(PIN_crosswalkBoard , status):
7         digitalWrite(PIN_crosswalkBoard , nivel = gpio.HIGH if status[0]
      else gpio.LOW)
8
9 def spotlight(PIN_spotlight, status):
```

```
10          digitalWrite(PIN_spotlight, nivel = gpio.HIGH if status[3] else
    gpio.LOW)
11
12  def pedestrianBoard(PIN_pedestrianBoard, status, timeCross):
13          digitalWrite(PIN_pedestrianBoard, nivel = gpio.HIGH)
14          time.sleep(tempoPass)
15          digitalWrite(PIN_pedestrianBoard, nivel = gpio.LOW)
16
17  def flashLight(PIN_flash, status):
18          digitalWrite(PIN_flashLight, nivel = gpio.HIGH if status[1] else
     gpio.LOW)
```

**codDecodPackage.py**

```
1  from lib.registers import *
2
3  def decPackage(msg):
4          msg = str(msg)
5          finalPackage = str()
6          finalPackage = modMsg(msg).split(",")
7          finalPackage = [finalPackage[0], True if finalPackage[1] == '1'
    else False, int(finalPackage[2])]
8          return finalPackage
9
10 def codPackage(idSmartcrosswalk, sensorPair, id_package):
11         sensorPair = int(sensorPair)
12         codedPacote = idSmartcrosswalk + "," + str(sensorPairr) + "," +
    str(id_package)
13         return codedPackage
```

**confPins.py**

```
1  import RPi.GPIO as gpio
2
3  gpio.setmode(gpio.BCM)
4
```

```
def pinMode(*pins, mode):
    for value in pins:
            gpio.setup(value, mode)

def digitalWrite(*pins, value):
    for value in pins:
            gpio.output(value, level)
```

### C.1.4 Tools

#### ___init___.py

```
import csv

def read():
        file = open('devicesID.csv', 'r')
        devicesID = list()
        try:
            reader = csv.reader(file)
            for line in reader:
                    devicesID.append([line[0], int(line[1])])
            # devicesID[type 'str', type 'int']
            return devicesID
        except:
            print("File's read Error!")
        finally:
            file.close()

def modMsg(ID):
    newID = str()
    ID = str(ID)
    ID = ID.replace("'"," ", 2)
    ID = ID.replace("b"," ",1)
    for element in ID:
            if element != " ":
                    newID = newID + element
```

```
25      return newID
26
27  def idTester(ID):
28      ID = modID(ID)
```

## C.2 Structural

### C.2.1 Main Routine

```
1  #-------------------------------------------------------#
2  ## >>>>>>>>>>>>>         LIBRARIES        <<<<<<<<<<<<< ##
3  #-------------------------------------------------------#
4
5  # Delay to ensure the crotab initialization
6
7  import time
8  time.sleep(2) # Default value is t = 20s
9
10 from raspi import * # Raspberry Pi module sets and operations' scripts
11 from communication import * # LoRa module sets and operations' scripts
12 from tools import * # Tools module's scripts
13
14 #-------------------------------------------------------#
15 ## >>>>>>>>>>>>>  GENERAL CONSIDERATIONS  <<<<<<<<<<<<< ##
16 #-------------------------------------------------------#
17
18 # 1) All OUTPUTS work as follow:
19 #      - HIGH VALUES [True/~3.3V] -> It is DESACTIVATED
20 #      - LOW VALUES [False/~0V] -> It is ACTIVATED
21 #      This logic is due the designed circuit configuration
22 # active the relays using LOW values to drive them.
23
24 # 2) The device type can be A or B, during all the algorithm
25 # the system make decisions based on it.
26
```

```
27
28 #-------------------------------------------------------------#
29 ## >>>>>>>>>>>>>     OBJECTS AND FILES    <<<<<<<<<<<<< ##
30 #-------------------------------------------------------------#
31
32 raspi = RaspberryPi(readDev(),readAuthorizedDev())
33
34 #-------------------------------------------------------------#
35 ## >>>>>>>>>>>>>       SET DEFAULTS       <<<<<<<<<<<<< ##
36 #-------------------------------------------------------------#
37
38 raspi._setIO() # It sets pins as IN or OUT
39
40 #-------------------------------------------------------------#
41 ## >>>>>>>>>>>>>      PRESET VALUES       <<<<<<<<<<<<< ##
42 #-------------------------------------------------------------#
43
44 # It sets all pin's OUT with default work values
45
46 raspi._allOutOff()
47
48 #-------------------------------------------------------------#
49 ## >>>>>>>>>>>>>         WORKPLACE        <<<<<<<<<<<<< ##
50 #-------------------------------------------------------------#
51
52 while True:
53
54     raspi._inputsDetection()
55
56     if raspi._status[2] or raspi._statusPair[2]: # Night detection [
    Night == True]
57
58         raspi._crosswalkBoard(True) # Crosswalk Board 'ON'
59
```

```
60          if raspi._status[0] or raspi._statusPair[0]: # Pedestrians on
    the region A' or B'
61
62              raspi._flashLight(True) # Flash Light 'ON'
63
64              if raspi._status[1]: # Detection of cars on the street [No
    movement == True]
65
66                  raspi._pedestrianBoard(True) # Allow the pedestrian
    crossing
67          else:
68
69              raspi._flashLight(False) # Flash Light 'OFF'
70              raspi._pedestrianBoard(False) # Pedestrian Board 'OFF'
71
72      else:
73
74          raspi._crosswalkBoard(False) # Crosswalk Board 'OFF'
75          raspi._flashLight(False) # Flash Light 'OFF'
76          raspi._pedestrianBoard(False) # Pedestrian Board
```

### C.2.2 Raspiberry Pi Module

**___init___.py**

```
1 from .raspi import *
```

**raspi.py**

```
1 #-----------------------------------------------------------#
2 ## >>>>>>>>>>>>>>>       LIBRARIES       <<<<<<<<<<<<<< ##
3 #-----------------------------------------------------------#
4
5 import RPi.GPIO as gpio # Library to control the GPIOS
6 import time
7
```

```
## >>>>>>>>>>>>>>>     SET LIBRARIES     <<<<<<<<<<<<<<< ##

gpio.setmode(gpio.BCM) # It sets default numeration to pins

#-------------------------------------------------------------#
## >>>>>>>>>>>>>>>        CLASSES        <<<<<<<<<<<<<<< ##
#-------------------------------------------------------------#

class RaspberryPi(object):

        def __init__(self, device=None, authorizedDevices=None): # It
   sets default information
            self._device = device
            self._authorizedDevices = authorizedDevices
            self._pinsIN = [27, 18, 19, 20]  # It sets default values to
    Raspiberry 3B+ INPUT pins
                        # = [SPI, SPI, 19, 20] - It sets default values
   to Raspiberry CM3 INPUT pins
                        # = [camera0,camera1,twilightSensor,
   obstacleSensor]
            self._pinsOUT = [23, 24, 25, 26] # It sets default values to
    Raspiberry CM3/3B+ OUPUT pins
                         # = [spotlight,flashLight,crosswalkBoard,
   pedestrianBoard]

            self._status = [False,False,False,False]
            self._statusPair = [False,False,False,False]

           # For status arrays = [camera',camera'',twilightSensor,
   obstacleSensor]

        def _pinModeIn(self): # It sets pin mode in
            for pin in self._pinsIN:
                        gpio.setup(pin,gpio.IN)

```

```
36        def _pinModeOut(self): # It sets pin mode out
37            for pin in self._pinsOUT:
38                        gpio.setup(pin,gpio.OUT)
39
40        def _setIO(self): # It sets predefined pins as IN or OUT
41            self._pinModeIn()
42            self._pinModeOut()
43            gpio.add_event_detect(self._pinsIN[3], gpio.RISING, callback
    =self._obstacleSensor,bouncetime=300)
44
45
46        def _inputsDetection(self):
47             self._camera0()
48             self._camera1()
49             self._twilightSensor()
50
51        def _allOutOff(self):
52             self._spotlight(False)
53             self._pedestrianBoard(False)
54             self._crosswalkBoard(False)
55             self._flashLight(False)
56
57        def _bounceTime(self,t=.1):
58             time.sleep(t)
59
60        def _camera0(self):
61            self._bounceTime()
62            if gpio.input(self._pinsIN[0]):
63                self._status[0] = True
64            else:
65                self._status[0] = False
66
67        def _camera1(self):
68            self._bounceTime()
69            if gpio.input(self._pinsIN[1]):
```

```
70                  self._status[1] = True
71              else:
72                  self._status[1] = False
73
74          def _twilightSensor(self):
75              self._bounceTime()
76              if gpio.input(self._pinsIN[2]):
77                  self._status[2] = True
78              else:
79                  self._status[2] = False
80
81          def _obstacleSensor(self,channel):
82              self._status[3] = True
83              while self._status[3] or self._statusPair[3] or self.
    _statusPair[1]:
84                  self._spotlight(True)
85                  self._status[3] = False
86              self._spotlight(False)
87
88          def _spotlight(self,command):
89             if command:
90               gpio.output(self._pinsOUT[0],gpio.LOW)
91             else:
92               gpio.output(self._pinsOUT[0],gpio.HIGH)
93
94          def _flashLight(self,command):
95             if command:
96                  gpio.output(self._pinsOUT[1],gpio.LOW)
97             else:
98               gpio.output(self._pinsOUT[1],gpio.HIGH)
99
100         def _crosswalkBoard(self,command):
101            if command:
102              gpio.output(self._pinsOUT[2],gpio.LOW)
103            else:
```

```
104             gpio.output(self._pinsOUT[2],gpio.HIGH)
105
106         def _pedestrianBoard(self,command):
107             if command:
108                 gpio.output(self._pinsOUT[3],gpio.LOW)
109             else:
110                 gpio.output(self._pinsOUT[3],gpio.HIGH)
```

### C.2.3 Tools

```
1  import csv
2
3  def readDev():
4          file = open('device.csv', 'r')
5          device = list()
6          try:
7              pointer = csv.reader(file)
8              for line in pointer:
9                      device = [line[0], int(line[1]), line[2], line[3]]
10                     # device = [type 'str', type 'int',type 'str', type
       'str']
11                     #          [id, channel, adress, deviceType]
12             return device
13         except:
14             print('ERROR to read the file!')
15         finally:
16             file.close()
17
18 def readAuthorizedDev():
19         file = open('authorizedDevices.csv', 'r')
20         devicesID = list()
21         try:
22             pointer = csv.reader(file)
23             for line in pointer:
24                     devicesID = [line[0],int(line[1])]
```

```
                    # devicesID = [type 'str',type 'int']
                    #           = [ID,channel]
            return devicesID
        except:
            print('ERROR to read the file!')
        finally:
            file.close()

def writeDev(ID):
        file = open('registeredDevices.csv', 'a')
        try:
            writer = csv.writer(file)
            writer.writerow(ID)
        except:
            print('ERROR to read the file!')
        finally:
            file.close()
```

## C.3 Communication

### C.3.1 Main Routine

```
#-------------------------------------------------------------#
## >>>>>>>>>>>>>         LIBRARIES        <<<<<<<<<<<<< ##
#-------------------------------------------------------------#

import RPi.GPIO as gpio
from raspi_lora import LoRa, ModemConfig
import time

#-------------------------------------------------------------#
## >>>>>>>>>>>>>     DEFAULT VALUES       <<<<<<<<<<<<< ##
#-------------------------------------------------------------#

PIN_twilightSensor = 19
```

```
14 PIN_spotlight = 23
15
16 pinMode(PIN_twilightSensor, mode = gpio.IN)
17 pinMode(PIN_spotlight, mode = gpio.OUT)
18
19 lora = LoRa(0, 17, 0, modem_config=ModemConfig.Bw125Cr45Sf128, tx_power
       =14, acks=True)
20
21 #-------------------------------------------------------------#
22 ## >>>>>>>>>>>>>          PRESETS          <<<<<<<<<<<<< ##
23 #-------------------------------------------------------------#
24
25 digitalWrite(PIN_twilighSensor, level = gpio.HIGH) # Ensure the output
       starting as LOW value
26 message = lora._last_payload # None value
27 status = False
28 flag = False
29
30 #-------------------------------------------------------------#
31 ## >>>>>>>>>>>>>        MAIN PROGRAM       <<<<<<<<<<<<< ##
32 #-------------------------------------------------------------#
33
34 while True:
35
36     lora.set_mode_rx()
37
38     try:
39         if bool(int(bytes.decode(lora._last_payload[0]))) is not message
       :
40                 message = bool(int(bytes.decode(lora._last_payload[0])))
41                 print(f"New message received!")
42     except:
43         pass
44
45     if bool(gpio.input(PIN_twilightSensor)) is not status:
```

```
        flag = True
        time.sleep(0.6)
        print(f"Status has changed")

    status = True if bool(gpio.input(PIN_twilightSensor)) else False

    if not(flag):

        if status or message:
                gpio.output(PIN_spotlight,gpio.LOW)
        else:
                gpio.output(PIN_spotlight,gpio.HIGH)

    else:
        lora.send('1' if status else '0',1)
        flag = False
```

## C.4 Integration

### C.4.1 Main Routine

```
#-------------------------------------------------------#
## >>>>>>>>>>>>>         LIBRARIES        <<<<<<<<<<<<< ##
#-------------------------------------------------------#

# Delay to ensure the crotab initialization

import time
time.sleep(20) # Default value is t = 20s

from raspi import * # Raspberry Pi module sets and operations' scripts
from communication import * # LoRa module sets and operations' scripts
from tools import * # Tools module's scripts

#-------------------------------------------------------#
```

```
15 ## >>>>>>>>>>>>>     OBJECTS AND FILES    <<<<<<<<<<<<< ##
16 #-----------------------------------------------------------#
17
18 raspi = RaspberryPi(readDev(),readAuthorizedDev())
19 lora = LoRa(readDev())
20
21 #-----------------------------------------------------------#
22 ## >>>>>>>>>>>>>       SET DEFAULTS       <<<<<<<<<<<<< ##
23 #-----------------------------------------------------------#
24
25 raspi._setIO() # It sets pins as IN or OUT
26
27 #-----------------------------------------------------------#
28 ## >>>>>>>>>>>>>      PRESET VALUES       <<<<<<<<<<<<< ##
29 #-----------------------------------------------------------#
30
31 raspi._allOutOff() # It sets all pin's OUT with default work values
32 lora.set_mode_rx() # It set the RFM95W as receptor
33
34 #-----------------------------------------------------------#
35 ## >>>>>>>>>>>>>         WORKPLACE        <<<<<<<<<<<<< ##
36 #-----------------------------------------------------------#
37
38 while True:
39
40     raspi._inputsDetection()
41     raspi._pairDetection(lora._message)
42
43     if not(raspi._flag):
44
45         if raspi._status[2] or raspi._statusPair[2]: # Night detection [
    Night == True]
46
47             raspi._crosswalkBoard(True) # Crosswalk Board 'ON'
48
```

```
49        else:
50             raspi._crosswalkBoard(False) # Crosswalk Board 'OFF'
51
52     else:
53         lora.send_to_wait(raspi._message(),raspi._authorizedDevices[1])
54         lora.set_mode_rx()
```

### C.4.2 Raspberry Pi Module Updates

```
1  def __init__(self, device=None, authorizedDevices=None): # It sets
       default information
2              self._device = device
3              self._authorizedDevices = authorizedDevices
4              self._pinsIN = [27, 18, 19, 20]  # It sets default values to
        Raspiberry 3B+ INPUT pins
5                       # = [SPI, SPI, 19, 20] - It sets default values
       to Raspiberry CM3 INPUT pins
6                       # = [camera0,camera1,twilightSensor,
       obstacleSensor]
7              self._pinsOUT = [23, 24, 25, 26] # It sets default values to
        Raspiberry CM3/3B+ OUPUT pins
8                        # = [spotlight,flashLight,crosswalkBoard,
       pedestrianBoard]
9
10             self._status = [False,False,False,False]
11             self._statusPair = [False,False,False,False]
12             self._flag = False
13
14 def _message(self):
15             strMsg = self._device[0]+','
16             for track in range(0,len(self._status)):
17                 if self._status[track] == True:
18                     strMsg = strMsg + '1'
19                 elif self._status[track] == False:
20                     strMsg = strMsg + '0'
```

```
            return strMsg

def _pairDetection(self,message):
    if message is not None:
    if self._authorizedDevices[0] == message[0]:
       if message[1] != self._statusPair:
       self._statusPair = message[1]
    else:
    pass

def _flagTest(self,pos):
   if self._flag:
    pass
   else:
    if bool(gpio.input(self._pinsIN[pos])) is not self._status[pos]:
      self._flag = True
        self._bounceTime(.2)
      else:
        self._flag = False
```

### C.4.3 LoRa Module Updates

```
class LoRa(object):
    def __init__(self,this_address):

        self._channel = 0
        self._interrupt = 17
        self._thisID = this_address[0]

        self._mode = None
        self._cad = None
        self._freq = 868
        self._tx_power = 14
        self._modem_config = ModemConfig.Bw125Cr45Sf128
        self._receive_all = False
```

```
14          self._acks = True
15
16          self._this_address = this_address[1]
17          self._last_header_id = 0
18
19          self._last_payload = None
20          self._message = None
21          self.crypto = AES.new(b'the_suffering_is_the_way')
22
23          self.cad_timeout = 0
24          self.send_retries = 3
25          self.wait_packet_sent_timeout = 0.2
26          self.retry_timeout = 0.2
27
28          # [...]
29          # The another setups of this class adaptation
30          # are defined equal to the original version.
31
32  def on_recv(self, payload):
33
34          msgDecod = list()
35          status = list()
36
37          try:
38             msgDecod.append(bytes.decode(payload.message).split(',')[0])
39             for value in bytes.decode(payload.message).split(',')[1]:
40                 if value == '1':
41                     status.append(True)
42                 else:
43                     status.append(False)
44             msgDecod.append(status)
45          except:
46              pass
```

## C.5 Final

### C.5.1 Main Routine

```
#-------------------------------------------------------------#
## >>>>>>>>>>>>>          LIBRARIES         <<<<<<<<<<<<< ##
#-------------------------------------------------------------#

time.sleep(20)

from raspi import *
from communication import *
from tools import *

#-------------------------------------------------------------#
## >>>>>>>>>>>>>      OBJECTS AND FILES     <<<<<<<<<<<<< ##
#-------------------------------------------------------------#

raspi = RaspberryPi(readDev(),readAuthorizedDev())
lora = LoRa(readDev())

#-------------------------------------------------------------#
## >>>>>>>>>>>>>        SET DEFAULTS        <<<<<<<<<<<<< ##
#-------------------------------------------------------------#

raspi._setIO()

#-------------------------------------------------------------#
## >>>>>>>>>>>>>       PRESET VALUES        <<<<<<<<<<<<< ##
#-------------------------------------------------------------#

raspi._outputCommand(SPOTLIGHT,\
                     FLASH_LIGHT,\
                     CROSSWALK_BOARD,\
                     PEDESTRIAN_BOARD,\
```

```
                    command=False)
lora._wait_devicePair(raspi._authorizedDevices[CHANNEL])

#-------------------------------------------------------------#
## >>>>>>>>>>>>>          WORKPLACE         <<<<<<<<<<<<< ##
#-------------------------------------------------------------#

while True:

    raspi._inputsDetection(CAMERA_0,CAMERA_1,TWILIGHT_SENSOR)
    raspi._pairDetection(lora._message)

    if not(raspi._flag):

        if raspi._status[TWILIGHT_SENSOR]\
        or raspi._statusPair[TWILIGHT_SENSOR]:

            raspi._outputCommand(CROSSWALK_BOARD,command=True)

            if raspi._status[CAMERA_0]\
            or raspi._statusPair[CAMERA_0]\
            or raspi._statusPair[CAMERA_1]:

                raspi._outputCommand(FLASH_LIGHT,command=True)

                if raspi._status[CAMERA_1]\
                or raspi._statusPair[CAMERA_1]:

                    raspi._outputCommand(PEDESTRIAN_BOARD,command=True)

                    if raspi._statusPair[CAMERA_1]:
                        raspi._outputCommand(SPOTLIGHT,command=True)
                    else:
                        raspi._outputCommand(SPOTLIGHT,command=False)

```

```
                else:
                    raspi._outputCommand(SPOTLIGHT,\
                                         PEDESTRIAN_BOARD,\
                                         command=False)

            else:
                raspi._outputCommand(SPOTLIGHT,\
                                     FLASH_LIGHT,\
                                     PEDESTRIAN_BOARD,\
                                     command=False)

        else:
            raspi._outputCommand(SPOTLIGHT,\
                                 FLASH_LIGHT,\
                                 CROSSWALK_BOARD,\
                                 PEDESTRIAN_BOARD,\
                                 command=False)
    else:
        lora.send_to_wait(raspi._message(),raspi._authorizedDevices[
    CHANNEL])
        lora.set_mode_rx()
```

### C.5.2 Raspberry Pi Module Updates

**constants.py**

```
#-------------------------------------------------------#
## >>>>>>>>>>>>>>>        CONSTANTS        <<<<<<<<<<<<<< ##
#-------------------------------------------------------#

## >>>>>>>>>>>>>>>          INPUTS         <<<<<<<<<<<<<< ##

# GPIO pins

PIN_CAMERA0 = 27
PIN_CAMERA1 = 18
```

```
11 PIN_TWILIGHT_SENSOR = 19
12 PIN_OBSTACLE_SENSOR = 20
13
14 # Status array's index
15
16 CAMERA_0 = 0
17 CAMERA_1 = 1
18 TWILIGHT_SENSOR = 2
19 OBSTACLE_SENSOR = 3
20
21 ## >>>>>>>>>>>>>>>          OUTPUTS         <<<<<<<<<<<<<<< ##
22
23 # GPIO pins
24
25 PIN_SPOTLIGHT = 23
26 PIN_FLASH_LIGHT = 24
27 PIN_CROSSWALK_BOARD = 25
28 PIN_PEDESTRIAN_BOARD = 26
29
30 # Pins array's index
31
32 SPOTLIGHT = 0
33 FLASH_LIGHT = 1
34 CROSSWALK_BOARD = 2
35 PEDESTRIAN_BOARD = 3
36
37 ## >>>>>>>>>>>>>>>      COMMUNICATION      <<<<<<<<<<<<<<< ##
38
39 # 'authorizedDevices.csv' index
40
41 ID = 0
42 CHANNEL = 1
43
44 # Received Package
45
```

```
STATUS_ARRAY = 1
```

**raspi.py**

```
#-----------------------------------------------------------#
## >>>>>>>>>>>>>>>       LIBRARIES       <<<<<<<<<<<<<< ##
#-----------------------------------------------------------#

import RPi.GPIO as gpio # Library to control the GPIOS
from .constants import *
import time

## >>>>>>>>>>>>>>>     SET LIBRARIES     <<<<<<<<<<<<<< ##

gpio.setmode(gpio.BCM) # It sets default numeration to pins

#-----------------------------------------------------------#
## >>>>>>>>>>>>>>>        CLASSES        <<<<<<<<<<<<<< ##
#-----------------------------------------------------------#

class RaspberryPi(object):

        def __init__(self, device=None, authorizedDevices=None): # It
   sets default
                                                                  #
   information
            self._device = device
            self._authorizedDevices = authorizedDevices
            self._pinsIN = [PIN_CAMERA0,\
                            PIN_CAMERA1,\
                            PIN_TWILIGHT_SENSOR,\
                            PIN_OBSTACLE_SENSOR]

            self._pinsOUT = [PIN_SPOTLIGHT, \
                             PIN_FLASH_LIGHT, \
```

```
30                              PIN_CROSSWALK_BOARD, \
31                              PIN_PEDESTRIAN_BOARD]
32
33          self._flag = False # Flag to detect any inputs' variation
34          self._status = [False,False,False,False]
35          self._statusPair = [False,False,False,False]
36
37         # For status arrays = [CAMERA_0, CAMERA_1, TWILIGHT_SENSOR,
    OBSTACLE_SENSOR]
38
39      def _message(self):
40          strMsg = self._device[ID]+','
41          for track in range(0,len(self._status)):
42              if self._status[track] == True:
43                  strMsg = strMsg + '1'
44              elif self._status[track] == False:
45                  strMsg = strMsg + '0'
46          return strMsg
47
48      def _pairDetection(self,message):
49          try:
50            if message is not None:
51              if self._authorizedDevices[ID] == message[ID]:
52                 if message[STATUS_ARRAY] != self._statusPair:
53                   self._statusPair = message[STATUS_ARRAY]
54          except:
55            pass
56
57      def _flagTest(self,pos,valueIn):
58          time.sleep(.1)
59          if valueIn is not self._status[pos]:
60             return True
61          else:
62             return False
63
```

```
64          def _pinModeIn(self): # It sets pin mode in
65              for pin in self._pinsIN:
66                          gpio.setup(pin,gpio.IN)
67
68          def _pinModeOut(self): # It sets pin mode out
69              for pin in self._pinsOUT:
70                          gpio.setup(pin,gpio.OUT)
71
72          def _setIO(self): # It sets predefined pins as IN or OUT
73              self._pinModeIn()
74              self._pinModeOut()
75
76          def _inputsDetection(self,*inputPins):
77              flag = list()
78
79              for inputPin in inputPins:
80                  valueIn = bool(gpio.input(self._pinsIN[inputPin]))
81                  flag.append(self._flagTest(inputPin,valueIn))
82                  self._status[inputPin] = True if valueIn else False
83
84              for change in flag:
85                  if change:
86                     self._flag = True
87                     break
88                  else:
89                     self._flag = False
90
91          def _outputCommand(self,*outputPins,command):
92               for outputPin in outputPins:
93                   gpio.output(self._pinsOUT[outputPin],gpio.LOW \
94                   if command else gpio.HIGH)
```

### C.5.3 LoRa Module Updates

```
1 def _wait_devicePair(self,devicePairChannel):
```

```
  flag = False
  while not(flag):
    flag = self.send_to_wait('',devicePairChannel)
    self.set_mode_rx()
    time.sleep(1)
```

## C.6 Setup Files' Generation Script

```
import random
import csv
import ipaddress

print('\n')
print('#################################################################')
print('#            SMART CROSSWALK - SETUP                            #')
print('#################################################################\n')

print('  This setup generate an ID and a Communication Channel, both')
print('randomics, for devices A and B. These values are saved in two ')
print("different .csv pairs of output files authorizedDevices_X.csv'")
print("and 'devices_X.csv, where X = A or B.\n")

print("Each pair of files must be copy and paste into their respective")
print("Devices on their Defaut Directory.\n")

print('Default directory: /home/pi/smartCrosswalk\n')

ID_A = str(ipaddress.IPv6Address(random.randrange(0,2**128,1)))\
       .replace("'","",2)
ID_B = str(ipaddress.IPv6Address(random.randrange(0,2**128,1)))\
       .replace("'","",2)

while ID_A == ID_B:
        ID_B = str(ipaddress.IPv6Address(random.randrange(0,2**128,1)))\
        .replace("'","",2)
```

```
28
29 channel_A = random.randrange(0,254,1)
30 channel_B = random.randrange(0,254,1)
31
32 while channel_A == channel_B:
33         channel_B = random.randrange(0,254,1)
34
35 address = str(input("Address: \n")).upper()
36
37 smartDeviceA = (ID_A, channel_A, address,'A')
38 smartDeviceB = (ID_B, channel_B, address,'B')
39
40         with open('device_A.csv','w') as file_A:
41                 writer = csv.writer(file_A)
42                 writer.writerow(smartDeviceA)
43
44         with open('device_B.csv','w') as file_B:
45                 writer = csv.writer(file_B)
46                 writer.writerow(smartDeviceB)
47
48         with open('authorizedDevices_A.csv','w') as file_A:
49                 writer = csv.writer(file_A)
50                 writer.writerow([smartDeviceB[0],smartDeviceB[1]])
51
52         with open('authorizedDevices_B.csv','w') as file_B:
53                 writer = csv.writer(file_B)
54                 writer.writerow([smartDeviceA[0],smartDeviceA[1]])
55
56         print('Files generated with success!')
```

# Appendix D

# Practical Protoboard Assembly

## D.1 Pilot

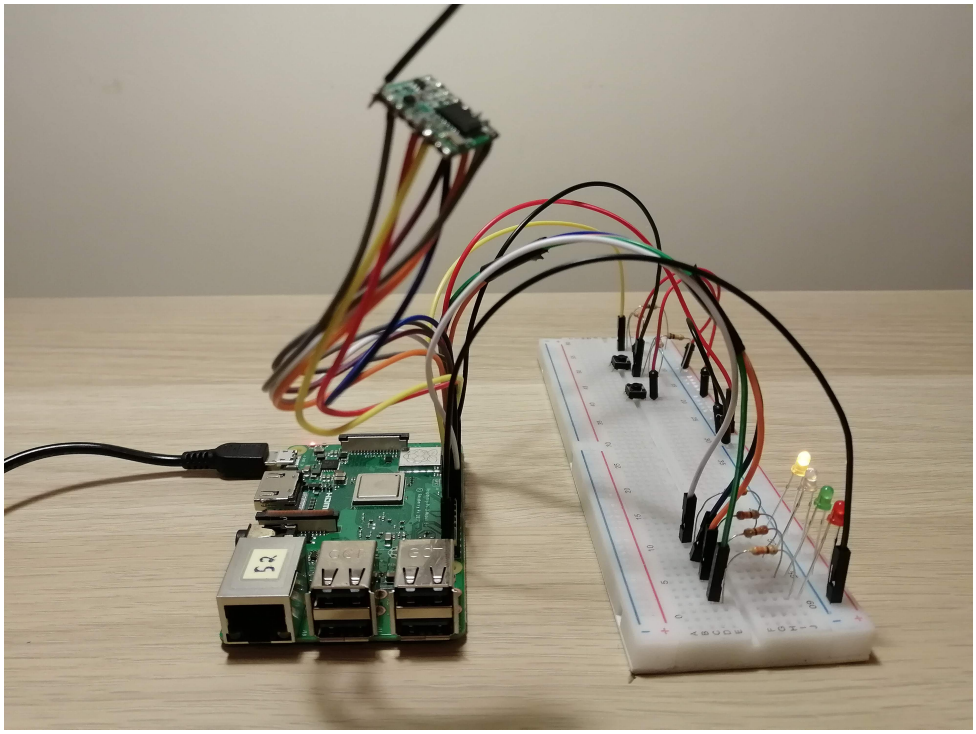


Figure D.1: General Pilot protoboard assembly.

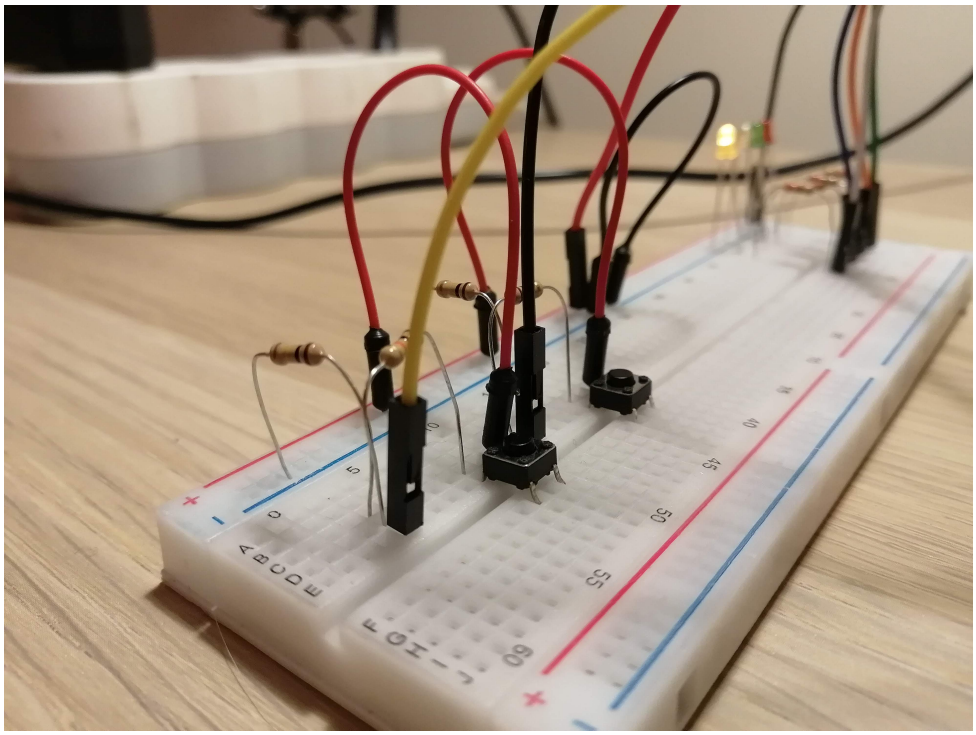


Figure D.2: Pilot inputs' protoboard assembly.

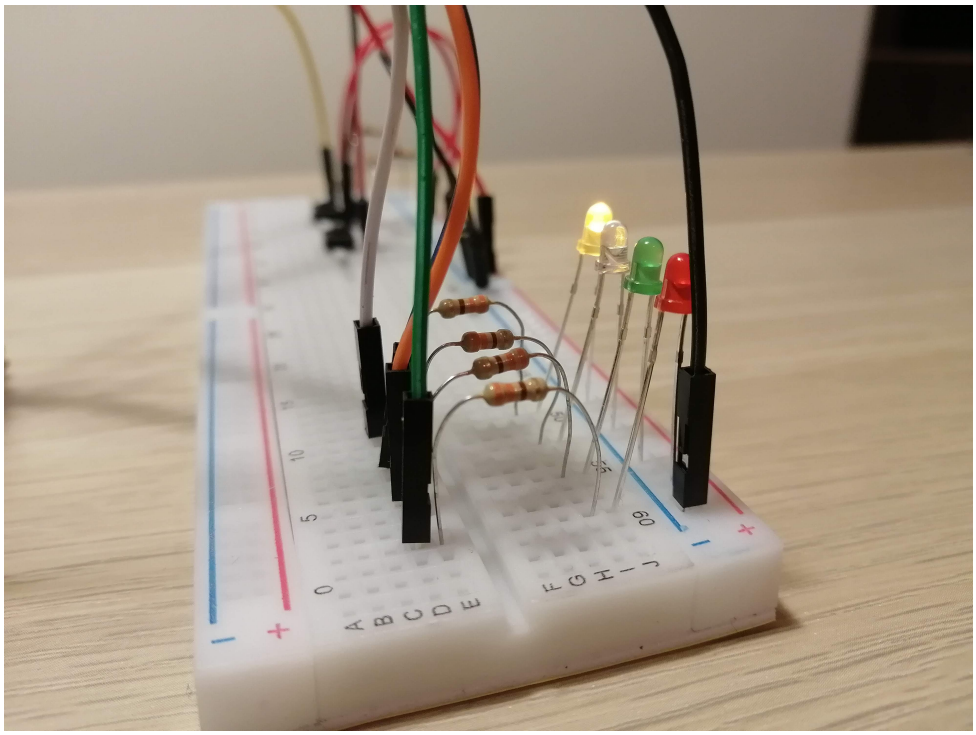


Figure D.3: Pilot outputs' protoboard assembly.

## D.2 Structural

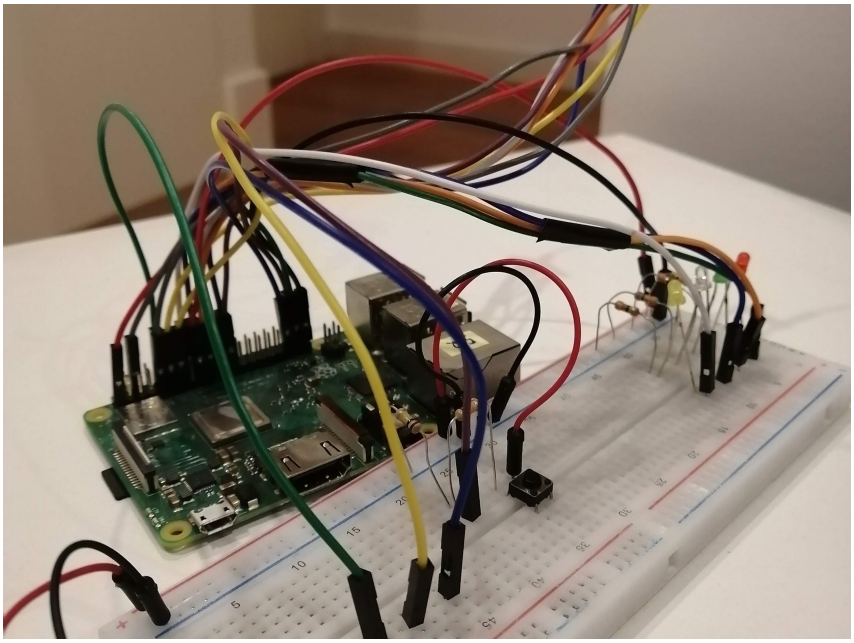


Figure D.4: General Structural protoboard assembly.

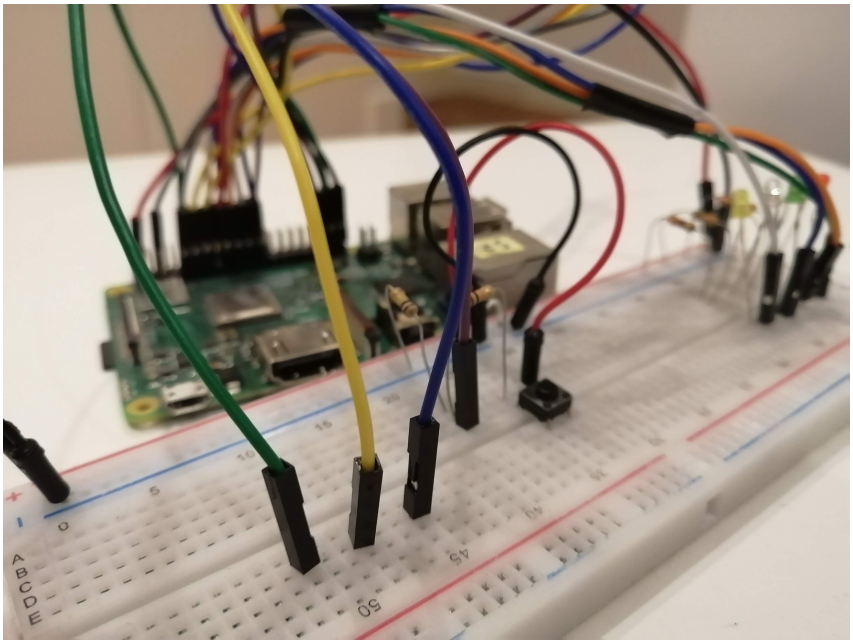


Figure D.5: Structural inputs' protoboard assembly.

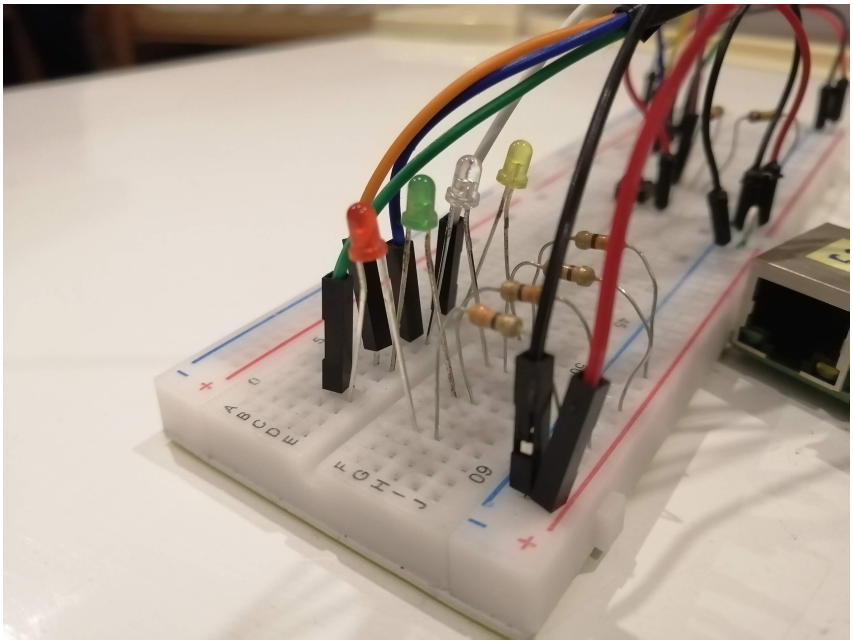


Figure D.6: Structural outputs' protoboard assembly.

## D.3 Final

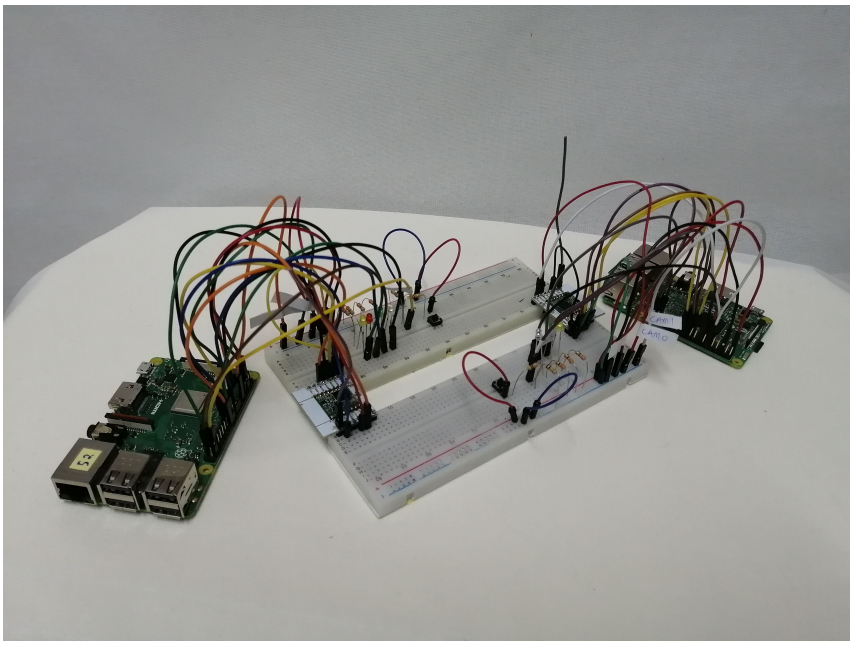


Figure D.7: General Structural protoboard assembly.

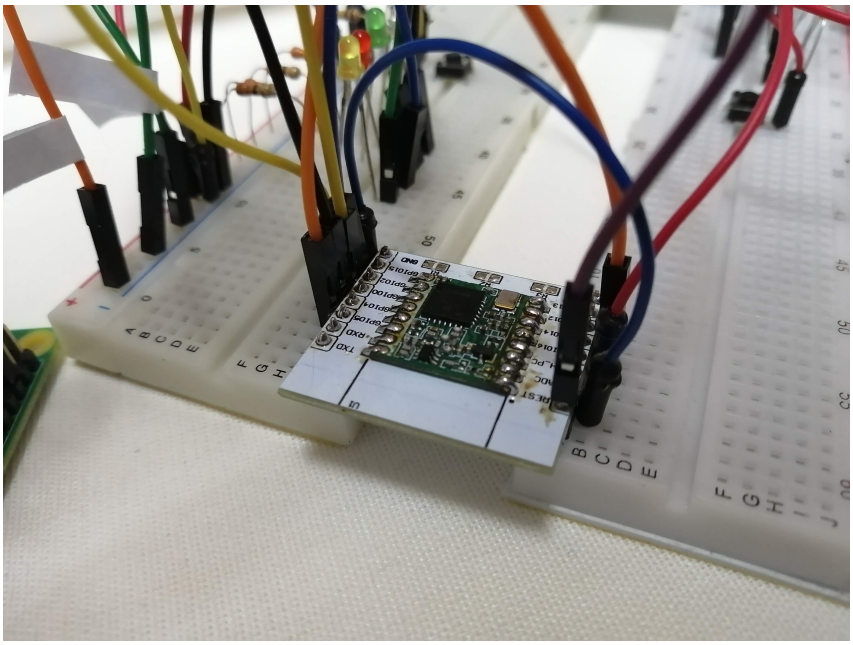


Figure D.8: Homemade RFM95W breakout.