TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Caner Pehlivan, 184594IVEM

NARROWBAND INTERNET OF THINGS BASED IMAGE SENSOR PLATFORM FOR SMART WASTE MANAGEMENT

Master's Thesis

Supervisor: Muhammad Mahtab Alam
Professor

Tallinn 2020

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Caner Pehlivan, 184594IVEM

KITSARIBALISEL ASJADE INTERNETI VÕRGUL PÕHINEV PILDISENSORI PLATVORM NUTIKAKS JÄÄTMEKÄITLUSEKS

Magistritöö

Juhendaja: Muhammad Mahtab Alam
Professor

Tallinn 2020

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Caner Pehlivan

18/05/2020

Abstract

Internet of Things (*IoT*) refers to the network of physical objects with internet connectivity, and the connection between these objects and other internet-enabled objects. Recently, it became a crucial part of our lives with the improvement of technology. IoT devices are everywhere offering solutions to make lives easier and efficient. Due to urbanization and rapid growth of population brought need for a smart waste management system especially for big cities. An IoT based solution can help cities to protect the environment and reduce costs.

The objective of this thesis is to create an efficient waste management system with the newest technologies possible and developing an accurate image processing algorithm for waste level detection compared to existing solutions. Additionally, establishing communication between the node and a server to analyse the collected data.

This system is aimed to overcome deficiencies of widely used sensors like the ultrasonic sensors in trash level measurement with a camera sensor. Also, one of the Low-Power Wide Area Network (*LPWAN*) communication technologies, Narrow Band – IoT (*NB-IoT*) is selected to establish communication between the sensor node and the cloud server, because of its global coverage and high data rate which will be used for image transmission.

This thesis proposes an image-based sensor platform that can be placed in the trash bins to identify the amount of trash. The platform consists of a camera module to take an image of the trash bin, and a communication module connected to the NB-IoT network to transfer the image. Then, an accurate grayscale level measurement algorithm is simulated and implemented. Accuracy of the algorithm reached up to %90, and the system can identify full trash bins successfully.

This thesis is written in English and is 56 pages long, including 6 chapters, 28 figures and 14 tables.

Annotatsioon

Kitsaribalise asjade interneti põhine kujutissensor nutikaks jäätmekäitluseks

Asjade interneti (*IoT*) all peetakse silmas internetiühendusega füüsiliste objektide võrku ning nende omavahelist- ja teiste internetti ühendatud seadmete vaheliste ühenduste. Asjade interneti näol on tegemist hiljutise, olulise lisandusega meie igapäevaelu mugavamaks muutmise suunas. IoT seadmeid võib leida kõikjal pakkumas võimalusi, kuidas elu lihtsamaks ja mugavamaks muuta.

Seoses linnastumise ja rahvastiku kiire kasvuga on tekkinud vajadust nutikaks jäätmekäitlemiseks, seda eriti just suurlinnades. IoT põhine lahendus aitab linnadel kaitsta keskkonda vähendada kulusid.

Käesoleva magistritöö eesmärgiks on luua efektiivne jäätmekäitlussüsteem kasutades selleks uusimaid tehnoloogiaid ja töötades välja täpse pilditöötlusalgoritmi prügi taseme tuvastamiseks võrreldes varasemate lahendustega.

Lisaks lahendada ka sensori ja serveri vahelise sideühenduse loomise probleem kogutud andmete analüüsimiseks. Süsteemi eesmärgiks on ületada laialt levinud ultrahelisensoritega prügi taseme mõõtmise probleemid kasutades kaamerasensorit. Valituks osutus madala võimsusega laivõrgu Low-Power Wide Area Network (*LPWAN*) Narrow Band – IoT (*NB-IoT*) sidetehnoloogia sensori ja serveri vahel kasutamiseks. Peamisteks valikukriteeriumiteks olid siin globaalne leviala ja, kujutise ülekandeks vajalik, suhteliselt suur andmeedastuskiirus.

Käesolevas töös pakutakse välja prügikasti paigaldatav pildisensorigasensorsüsteem, mis võimaldab tuvastada prügi hulka. Sensorplatvorm koosneb kujutist fikseerivast kaameramoodulist ja piltide serverisse edastamist võimaldavast NB-IoT sidemoodulist. Seejärel simuleeriti ja realiseeriti halltoonidel kasutav pilditöötlusalgoritm prügi taseme määramiseks. Algoritmi saavutatud täpsus oli kuni 90% ning süsteem suudab edukalt identifitseerida täis prügikastie.

Lõputöö on kirjutatud Inglise keeles ning sisaldab teksti 56 leheküljel, 6 peatükki, 28 joonist, 14 tabelit.

List of abbreviations and terms

3GPP	Third Generation Partnership Project
API	Application Programming Interface
APN	Access Point Name
BPSK	Binary Phase-Shift Keying
FTP	File Transfer Protocol
GLCM	Gray Level Co-occurrence Matrix
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HTTPS	Hypertext Transfer Protocol Secure
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineering
IoT	Internet of Things
KNN	K-Nearest Neighbours Algorithm
KPI	Key Performance Indicator
LoRa	Long Range
LPWAN	Low-Power Wide Area Network
LTE	Long Term Evolution
MLP	Multilayer Perceptron
MQTT	Message Queuing Telemetry Transport
NB-IoT	Narrowband Internet of Things
RFID	Radio-Frequency Identification
RGB	Red Green Blue
SIM	Subscriber Identity Module
SMS	Short Message Service
SPI	Serial Peripheral Interface
SQL	Structured Query Language
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver-Transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus
VGA	Video Graphics Array

Table of contents

List of figures

List of tables

1 Introduction

Over the years, waste management has become a major problem due to rapid urbanization. Populations are increased in cities as well as the amount of garbage [1] . Thus, the complexity of waste management occurred especially in megacities which have countless trash bins. Traditionally, garbage collection trucks collect all the waste by visiting each waste bin one by one. An unorganized garbage collection system like this results in overflowing waste bins and extra costs for waste collection trucks. These overflowing bins cause many other negative environmental impacts such as air, soil, and water contamination, pests, and bacteria growth. Thus, urban cities need to have an optimized system to observe and to collect all the garbage generated.

With the new developments in technology, cities are becoming smarter [2]. Those smart cities aim to reduce different costs and increase efficiency as well as the life quality of its inhabitants by having smart solutions combined with technology. This technology is nothing but the Internet of Things (IoT), a widely known phenomenon over the last years. IoT devices started to take part in our daily lives, sometimes as a smartwatch, smart thermostat, or in agriculture. According to statistics, there will be 50 billion connected devices worldwide in 2030 [3], and some of these devices will belong to smart waste management systems.

Waste management starts with measuring the amount of the trash inside a waste bin. Ultrasonic and load sensors are commonly used for this purpose in different ways. However, camera sensors are not widely preferred due to high costs and lacking image processing algorithms for trash level measurements. Nonetheless, technology now starts to offer low-cost camera modules for IoT systems. This brings out the motivation behind this thesis to propose an accurate image processing method and use a camera module for this purpose.

The last stage of waste management is informing responsible organizations with the status of the trash bin. So, waste collection can be optimized according to the data collected by sensors. Most of the current waste management systems use communication technologies such as Wi-Fi, GSM, or RFID. Yet, the rapid growth of IoT market increased demand for Low-Power Wide Area Network technologies like NB-IoT and LoRa [4]. They offer strong characteristics over other communication technologies in terms of power consumption, latency, and cost. These characteristics are the main motivation behind this thesis to use LPWAN technologies and tackle smart waste management problem.

This thesis work is aimed to propose a robust IoT system for waste management. In Chapter 1, a brief introduction and problem statement are represented. Chapter 2 describes the State of the Art on waste management. Chapter 3 presents the proposed system by explaining hardware used in this thesis, possible network interface, and cloud server solutions. In Chapter 4, a new image processing algorithm is proposed, and simulation results are discussed. Chapter 5 describes the implementation considerations of the proposed IoT system. Lastly, Chapter 6 concludes the thesis and discusses improvement opportunities as future work.

1.1 Problem Statement

Smart waste management system needs to provide proper garbage tracking and should consistently alert responsible organizations. This thesis focuses on establishing the system using a camera inside of the trash bin and sending the data to the cloud using NB-IoT. Moreover, data will be analysed using image processing algorithms, and trash will be collected by the municipality truck only if it is needed. To sum all up, this thesis will focus on solving three important problems:

- Creating an efficient waste management system.
- Developing an accurate image processing algorithm for waste level detection compared to existing solutions.
- Establishing a communication between the node and a server to have more precise decisions when collecting the waste with decreased latency.

2 State of the Art

In this section, current smart waste management systems are presented and discussed in three parts. Firstly, current trash level measurement sensors such as ultrasonic, load and camera sensors are analysed. Secondly, commonly used communication technologies are described and compared in terms of important KPI's such as latency and data rate. Lastly, an overview of recent smart waste management studies and solutions are given in a table with comments.

2.1 Level Measurement Methods

One of the key elements of IoT is sensor nodes which are responsible for collecting some data and sending it to a connected environment to further data analysing. These sensor nodes usually consist of a microcontroller unit, a connectivity unit, and one or more sensors. In order to create a smart waste management system using IoT, the amount of waste in the trash bin must be measured. Thus, a level measurement method must be decided. There are many different works in this context for level measurement, using different sensors. In this section, commonly used sensors are analysed, and their drawbacks are discussed.

2.1.1 Ultrasonic Sensors

Ultrasonic sensor is a device used for measuring the distance between itself and an object. They are working with a simple logic: They emit sound waves from their transmitter and wait for the sound to be reflected, and calculate the distance based on the time. This sensor can provide continuous information about the trash level in the dustbin by measuring the trash distance. Sensor shown in Figure 1 is a widely used ultrasonic sensor model (HC-SR04) and it has 4 pins, Trig, Echo, Vcc, and GND. Trigger (Trig) pin must be triggered using 10µs pulse, and Echo pin will detect the reflected pulse to measure the distance. Range of this specific model is between 2cm to 400cm. They are commonly used in smart waste management research papers and solutions to measure the distance between top of the trash bin and the waste. Some shortcomings of ultrasonic sensors are:

- Depending on type of the trash bin, they need to be mounted under the lid, and often those lids may be left open.
- Sensing accuracy can be affected by soft materials and liquids. Ultrasonic sensors work best towards solid materials, and sometimes trash bin include different type of items. [5]
- Sometimes there are so many echo signals reflected, and some sort of filter needs to be used to enhance quality of the data. [5]
- Ultrasonic sensors highly depend on container type because of their measurement angles. HC-SR04 ultrasonic sensor has 30 degrees measurement angle, and in case of uneven load in a trash bin sensor gives false alarm as shown in Figure 2.
- With questionable accuracy ultrasonic sensors also does not provide a visual confirmation for the trash bin. In case of false alarms, there is not any way to check actual waste and confirm the alarm.

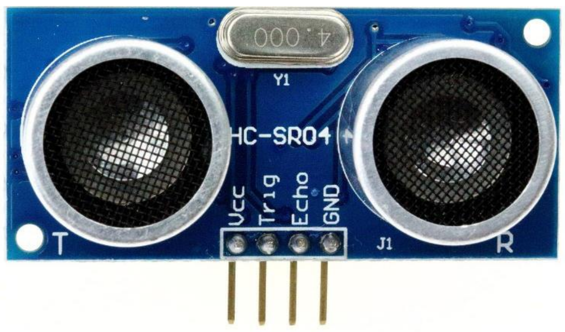


Figure 1. HCSR-04 Ultrasonic Sensor.

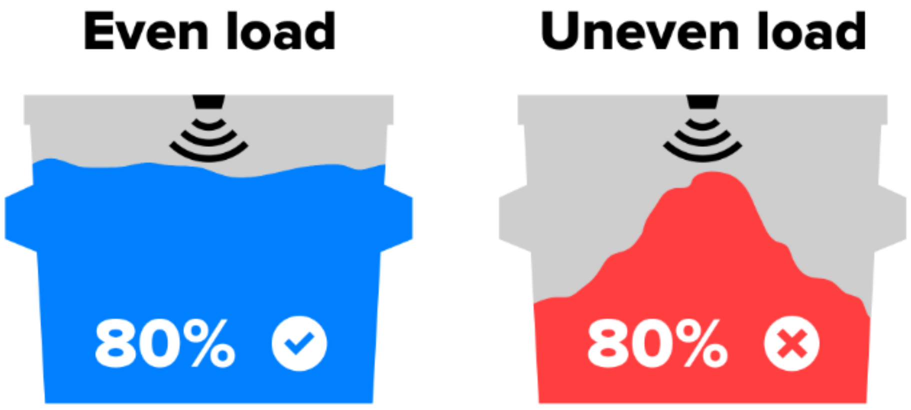


Figure 2. Even load and uneven load comparison for wide trash bins. [6]

2.1.2 Load Sensor

The load cell measures force and gives this force as an electric signal. Different types of load cells are hydraulic load cells, pneumatic load cells and strain gauge load cells. It can be placed below of the trash bin and measure weight of the trash. One drawback about this sensor is electrical signal output is usually in millivolts and sensor requires an amplifier as shown in Figure 3 to get measurable data [7]. This sensor is used in many different projects and even though it is a relatively cheap sensor, it needs to be combined with another sensor for level measurement because some type of thrash can cover more space in thrash bin but can be light. Thus, this solution unlikely to use alone, it must be supported by other sensors.

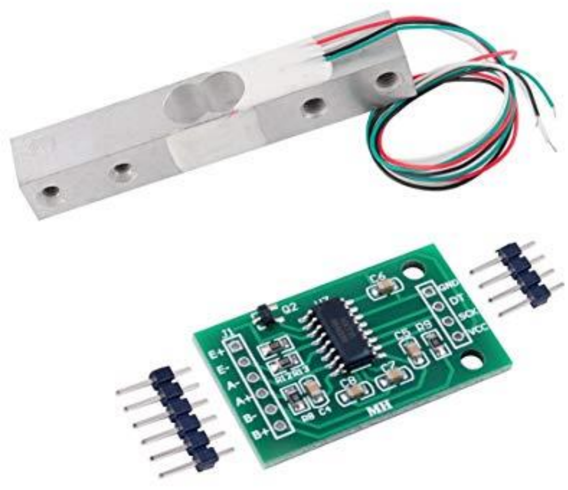


Figure 3. Load sensor and amplifier circuit.

2.1.3 Camera Sensors and Image Processing

A low-cost camera is also used as a level measurement method in several projects by placing the camera inside of the trash bin. There are two methods to use image processing algorithms for level measurement. One of the methods is transferring a taken image to a cloud network. Afterward, this image can be processed with image processing techniques and a more precise result can be derived compared to ultrasonic and load sensors, using estimation algorithms [8]. Another method is processing the image on the node, and deciding whether the bin is full or not, and only sending the alert signal. Even though the camera requires large throughput because of the enormous image data, accuracy is higher than other methods. Also, the camera method has a visual confirmation opportunity for the responsible organizations. So, the risk of false alarms is less compared to any other system.

2.2 Communication Technologies

Another key element of every IoT system is a communication technology to transfer the data collected by sensors. This transfer can be between two or more nodes, or between a node and a cloud environment. Data can be used for different purposes such as analysing and decision-making. In case of smart waste management systems, different communication technologies such as GSM, Wi-Fi, Bluetooth, and ZigBee are widely used. In this section, overview of such communication technologies and their pros and cons are discussed in sub-sections.

2.2.1 GSM

GSM (Global System for Mobile Communication) is a mobile network widely used by mobile phone users in the world. This standard is developed by the European Telecommunications Standards Institute (ETSI) in 1982. Now, it is commonly used for mobile voice transmission. Operation frequency of GSM is 900 MHz and 1800 MHz, depending on the country [9]. Data rates are between 64 kbps to 120 Mbps. GSM is used in many different works in scope of waste management systems.

In proposed works, SMS capability of GSM is used as an alert signal to the responsible organizations when a trash bin is full. Drawback of GSM is high running costs [10], SMS capability is needed for sending SMS and SIM card needs to be registered to the network on the node. Which means additional price for the telecom companies. Moreover, countless number of trash bins in big cities motivates us to bring low cost and low power consumption solutions.

2.2.2 Wi-Fi

Wireless connectivity is known as Wi-Fi which allows us to connect our devices to the internet without a physical wire. It is a short-range wireless communication technology which widely used in homes, offices, and mostly everywhere except outdoor places. The official name and standard for the Wi-Fi is IEEE 802.11 which is released in 1997 [11]. Currently, Wi-Fi consists of more than 20 different standards, which are developed versions of the first release. These standards are represented with a letter added to the end of the name. There are two different standards widely known and used and they operate in 2.4 GHz band, with maximum 54 Mbps data rate and 250m coverage area [12]. Wi-Fi technology is broadly used in different waste management works. Most of the proposed solutions are in-door use cases and in case of rural areas, Wi-Fi may be inaccessible and a limited technology due to modem requirement. It is easily not suitable for waste management solutions.

2.2.3 Bluetooth

Bluetooth is another wireless technology standard, mainly used for short range data exchange. It is a peer-to-peer communication technology which eliminates physical cable connection requirement for data transmission. Standard of Bluetooth is IEEE 802.15 and it is released in 2003. Operating frequency of the Bluetooth is 2.5 GHz ISM band [13]. Bluetooth's maximum coverage range can be extended up to 100 meters. It can be used in indoor environments to send an alert to the responsible person of trash bin in case it is overflowing. However, because of its short range it is not possible to use Bluetooth as a communication technology in outdoor waste management systems.

2.2.4 ZigBee

ZigBee is an open global standard for low-data rate and low-power applications. It is based on the IEEE 802.15 specification and developed by ZigBee Alliance. ZigBee wireless power area networks operate on 2.4 GHz, 900 MHz and 868 MHz frequencies in license free bands [14]. Over air data rates of ZigBee is up to 250kb/s, and it is widely used in waste management projects which requires low throughputs. ZigBee has a direct range limited to 15-20 meters in open air, which is extremely limited for the data transmission with smart bins over a city. ZigBee also can be used in in-door applications for waste management systems, but it is not applicable in outdoor systems. Range can be increased by building a mesh network, but large interconnections are required which increases complexity of the system. [15]

2.2.5 LPWAN

LPWAN technologies are types of wireless telecommunication wide area networks designed to be used in long-range communications at a low bit rate. LPWAN technologies have three main benefits for a smart waste management system like this project compared to the others:

- **Long Range:** Depending on the technology, they have a few kilometers range in urban areas, and can be used in trash bins located in far away.

- **Low Power:** They are optimized for power consumption and inexpensive batteries can be used up to 20 years.
- **Low Cost:** LPWAN does not require high quality hardware designs due to the reduced complexity.

Three important and widely used LPWAN technologies are SigFox, Long Range (LoRa), and NB-IoT [3].

2.2.5.1 SigFox

SigFox is a LPWAN technology developed in 2010 in France [16]. It operates in unlicensed spectrum meaning that it is not a cellular technology. These unlicensed bands are 915 MHz in North America, 868 MHz in Europe, and 433 MHz in Asia. SigFox sets up base stations to allow data transmission for its devices [17]. Its technology is designed to send small amounts of data (12 bytes) using standard radio transmission method binary phase-shift keying (BPSK) in an ultra-narrow band (100Hz) carrier. SigFox offers low energy consumption and cost efficiency like all LPWAN technologies. However, it is utilized for small messages meaning that large assets or media like an image is not allowed on the network.

2.2.5.2 LoRa

LoRa(also referred as LoRaWAN) is another LPWAN technology designed for M2M and IoT applications by LoRa Alliance [18]. It uses same unlicensed ISM bands like SigFox. Its technology is based on chirp spread spectrum (CSS) modulation which spreads a narrow-band signal on a wider channel bandwidth [3]. Thus, the transmitted signal has low noise levels. LoRa has a data rate is between 300 bps and 50 kbps and range up to 20 kilometres in rural areas. Even though its impressive range, LoRa is not a cellular technology and requires multiple base stations depending on the application. This may increase network deployment cost.

2.2.5.3 NB-IoT

NB-IoT is a radio technology standard introduced in Release 13 of the 3GPP (3rd Generation Partnership Project) [19].It is classified as a 5G technology in 2016 [20]. Unlike SigFox and LoRa, NB-IoT is a licensed cellular technology and can operate under GSM and LTE (long-term evolution) in frequency bands such as 700 MHz, 800 MHz and 900 MHz, by occupying 200 kHz frequency bandwidth [21]. NB-IoT has three different operation modes in LTE and GSM carriers as shown in Figure 4:

- In-band mode: utilizing a resource block within an LTE carrier.

- Guard-band mode: utilizing the guard band resource blocks of LTE for NB-IoT carrier.
- Stand-alone mode: replacing one or more GSM carriers for NB-IoT.

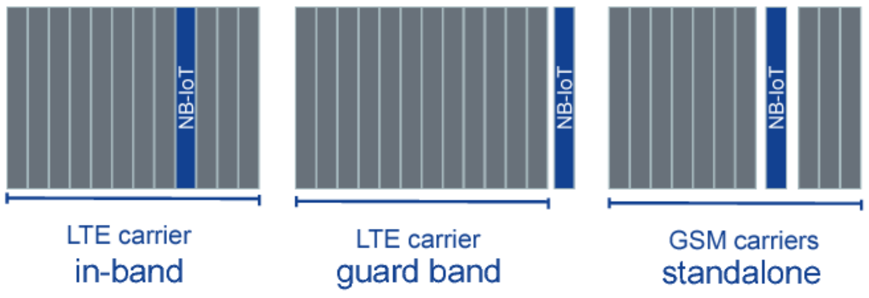


Figure 4. NB-IoT operation modes [22]

NB-IoT can be deployed in existing LTE infrastructure by only a software update. Which brings a wide coverage and flexibility. Cities are full of LTE base stations, providing cellular communication like 4G to the citizens. Especially in smart waste management domain, ability to use LTE base stations is extremely important. In Release 13 of 3GPP, NB-IoT is characterised as shown in Figure 5:

- Excellent indoor coverage of 20dB,
- Massive number of low-throughput device support (52547 devices in a cell-site sector),
- Cost efficiency with low complexity devices,
- Low power consumption like all LPWAN technologies,
- Latency lower than 10 seconds or less.

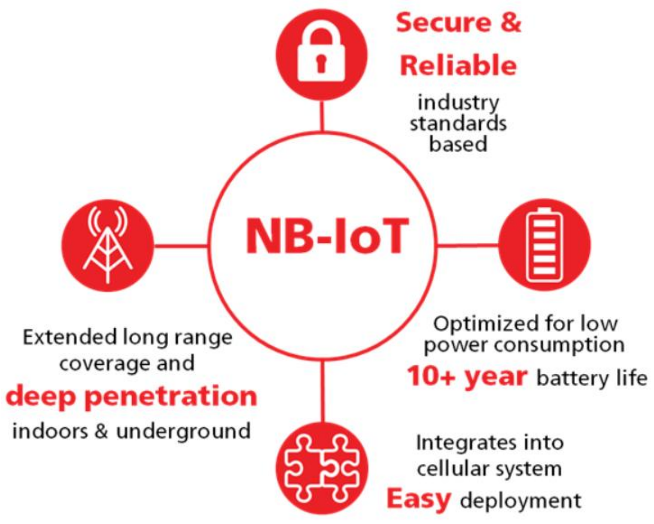


Figure 5. Benefits of NB-IoT [23]

Table 1 indicates the comparison between technical details of SigFox, LoRa and NB-IoT. First, the most important distinction between these LPWAN technologies are licensed and unlicensed spectrum. Support of already existing and required infrastructure minimizes the costs required for deployment of IoT systems for NB-IoT. However, SigFox is not available everywhere and LoRa needs local network deployment. In terms of network coverage and range, NB-IoT is limited with LTE base stations. Ericsson predicted that the global population coverage of LTE networks would increase from 75% to over 90% in 2025 [24]. Thus, NB-IoT coverage is the most promising one among others for a smart waste management system deployed in a city. Another important aspect is data rates. Size of an 8-bit grayscale image taken by camera sensor in 320x240 resolution can be calculated as 75 kB (by multiplying pixels in width and height and 8-bit depth level). However, this size can change depending on compression, colour and resolution of the image. Among all three technologies, NB-IoT is capable to send large data sizes with highest data rate [3]. Security is not considered as a KPI in this thesis, nevertheless NB-IoT already operates in an existing secure network of 3GPP. Battery life for each technology is already over 10 years. Lastly, module cost of NB-IoT is less than LoRa but SigFox is more cost-effective compared to NB-IoT. To sum all up, NB-IoT stands out with its licensed cellular band and data rate support for image transmission over a network.

Table 1. Comparison of LPWAN technologies [25].

	SigFox	**LoRa**	**NB-IoT**
Spectrum	Unlicensed	Unlicensed	Licensed
Bandwidth	100 Hz	125 kHz	180 kHz
Coverage	149 dB	157 dB	164 dB
Data Rate	100 bps	50 kbps	250 kbps
Scalability	Low	Medium	High
Security	16 bit	32 bit	3GPP (128-256 bit)
Battery Life	10+ years	10+ years	10+ years
Module Cost	2$	12$	5$

2.3 Analysis of Existing Works

In this section, some of the existing solutions for solid waste management systems are given in Table 2, including measurement methods and communication technologies used. Also, drawbacks and comments about each paper is given.

Table 2. Analysis of different waste management solutions.

Reference	**Sensor(s)**	**Communication Technology**	**Microcontroller**	**Methodology and Limitations**
[26] (2017)	Ultrasonic	Wi-Fi	-	• Waste truck routes are optimized with sensor data. • Wi-Fi is a short-range technology and not available in the rural areas.
[2] (2018)	Ultrasonic	Wi-Fi, GPS	Arduino Uno	• GPS module for location information • Wi-Fi range
[7] (2017)	Ultrasonic Humidity Load Cell	Wi-Fi	Raspberry Pi 3	• Load cell sensor combined with ultrasonic • Humidity sensor to distinguish between dry and wet waste • Truck driver informed with a web GUI
[27] (2017)	Ultrasonic Load Cell	RF GSM Module	PIC16F877A Arduino	• Two different sensors for better accuracy • RF transmitter used with GSM module • Too much equipment for one bin

[28] (2017)	Ultrasonic	GSM Module LED(s)	Arduino Uno	• Green LED to alert all the residents • Red LED if garbage overflows • Only acceptable for apartments
[8] (2012)	Camera	RFID GIS GPRS	-	• RFID tag for bins • A camera is attached to the truck • Images before and after collecting the trash • Waste level is estimated using images • GLCM feature extraction method is used for the image processing.
[29] (2015)	Ultrasonic	Ethernet or Wi-Fi	-	• Range of Wi-Fi • Accuracy of the ultrasonic sensors
[30] (2017)	Ultrasonic	GSM	Node MCU	• Notification via GSM technology
[31] (2016)	Infrared Load Cell	Wi-Fi	ARM LPC2148	• IR sensor combined with load cell • Range of Wi-Fi
[32] (2017)	Ultrasonic Load Cell	GSM	-	• Notification via GSM technology
[33] (2017)	Ultrasonic	Wi-Fi	Arduino Uno	• Notifies the municipality instead of truck driver

				• Range of Wi-Fi
[34] (2016)	Ultrasonic	ZigBee GSM	Arduino Uno	• ZigBee and GSM for notification • ZigBee range • High running costs of GSM
[35] (2014)	Ultrasonic	ZigBee GSM	UARP microcontroller	• ZigBee range • GSM costs • System notifies the municipality instead of truck driver.
[36] (2017)	Ultrasonic	RFID	-	• RFID reader for citizens • Citizens interact with trash bin • Problem in case of forgetting the card

3 Proposed System

Previously in the Introduction chapter, it is mentioned that major components of IoT are sensors, gateways, cloud, analytics, and user interface. This chapter introduces the components of the proposed IoT system to tackle waste management problem. The basic system architecture is shown in Figure 6.

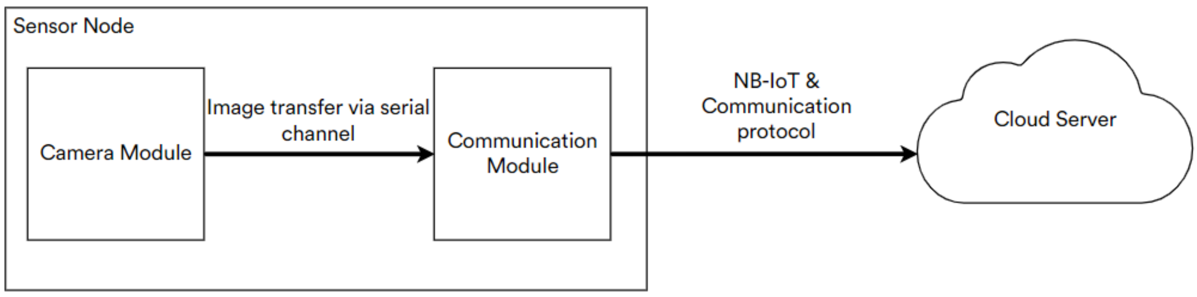

Sensor Node
Camera Module
Image transfer via serial channel
Communication Module
NB-IoT & Communication protocol
Cloud Server

Figure 6. Basic architecture diagram of the proposed system.

One can see that the proposed system consists of four main parts:

- Camera module
- Communication module
- Network interface
- Cloud server

The camera module is responsible to measure the level of trash in trash bin. Afterward, a communication module is necessary to establish communication between the sensor node and the cloud server. Camera and communication modules must be compatible with each other to transfer images between themselves. This transfer of data can be possible using data transfer interfaces such as UART, I2C, or SPI. Afterward, transferred data needs to be sent to a cloud server from the communication module with a network protocol. There are different network protocols available for this purpose in both the transport layer and application layer, and they are discussed briefly in this chapter. Lastly, a cloud server is required to store data and further analyse it. Possible solutions for backend setup are explained in the Backend subsection.

3.1 Hardware

In this subsection, details about hardware modules used in this thesis are discussed. First, widely used camera sensors for IoT devices are introduced and compared. Second, communication module used in this project explained in detail.

3.1.1 Camera Module

In recent years, camera modules became an important part of IoT sensors. These modules became smaller and cheaper as technology improves and used in many different areas of IoT such as security systems, smart vehicles, smart home applications, and wildlife imaging solutions like forest traps. There are some important low-cost IoT camera modules in the market to use in IoT projects. These cameras include Pi Camera, Arducam, and OpenMV etc.

The Raspberry Pi Foundation developed the Pi Camera Module v2 in April 2016 [37]. It is a camera module with an improved Sony IMX219 8-megapixel camera sensor compared to the previous model. Pi Camera Module v2 allows user to capture images and videos in 1080p30, 720p60, and VGA90 video modes. Pi Camera Module v2 works with Raspberry Pi models connected via Camera Serial Interface (CSI) port. Raspberry Pi is a single-board computer with different input and output ports, ethernet support, and an ARM Cortex processor. These specifications are not necessary for a smart waste management application as high-quality images of trash bin are not required for the level measurement algorithm. Thus, Pi Camera Module v2 is not used in this thesis.

Arducam is another company manufacturing low-cost IoT cameras for Arduino and Raspberry Pi [38]. The company also offers customized designs for the customers. They introduce cameras in three categories, Serial Peripheral Interface (SPI) cameras, Raspberry Pi cameras, and Universal Serial Bus (USB) cameras. SPI camera products are introduced in 2012 as a general-purpose solution for Arduinos, however, it is not limited to the Arduino platform and can be used with other hardware through SPI and I2C interfaces. Arducam also offers camera modules compatible with Raspberry Pi to support people who want to build their projects on Raspberry Pi. Lastly, USB cameras are introduced in 2017 by Arducam which offers much higher resolutions compared to low-cost IoT cameras, but they contradict IoT in terms of cost-efficiency. In the early stages of this thesis, the Arducam OV5642 mini module considered as a camera module as shown in Figure 7. OV5642 is a 5MP general-purpose SPI camera supplied by 3.3V~5V and supports 1080p, 720p, VGA, QVGA resolutions and RAW, YUV, RGB, JPEG image formats. However, this camera is developed by Arducam to be used with an Arduino board which is a single-board microcontroller. OV5642 is tested with STM32L476RG microcontroller instead of an Arduino board because Nucleo boards offer higher clock frequencies and flash memories. STM32L476RG is an ultra-low-power microcontroller with Arm Cortex-M4 core and can be used in different IoT applications [39]. After all, OV5642 must be configured with STM32 microcontroller via SPI and I2C busses, but it is not a practical solution for a thesis because of complexity of establishing communication between these two modules.

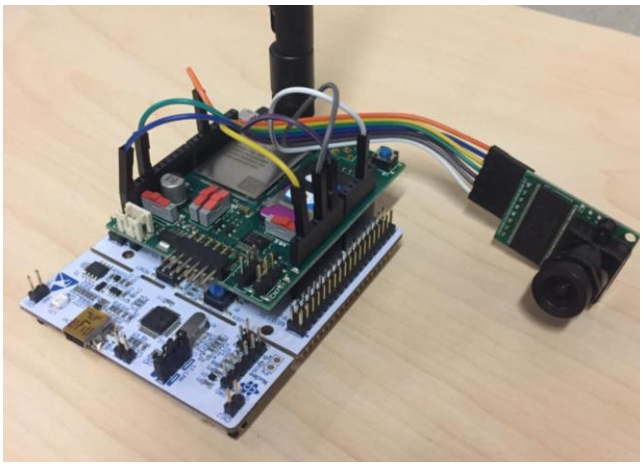


Figure 7. First prototype using STM32L476RG microcontroller (white module), Avnet BG96 Communication Shield (green module), and OV5642 Camera

OpenMV is a project started in 2013, in search of a better serial camera module [40] by Ibrahim Abdelkader. The goal was having a low cost, small form, and open-source camera module that can provide basic image processing applications for IoT projects. After making a boost with KickStarter, OpenMV became a company with camera solutions. In this thesis, OpenMV Cam H7 is used as a camera sensor shown in Figure 8.

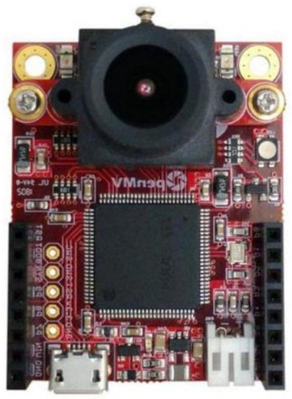


Figure 8. OpenMV Cam H7 Module [41].

An outstanding attribute of this module is it features a STM32H743VI ARM Cortex M7 processor running at 480 MHz and 1MB SRAM & 2MB of flash [41]. This allows users to program the OpenMV Cam without any extra microcontrollers. So, IoT applications can be implemented easily with only one module for a low price. The module has 3.3V~5V tolerant I/O pins and following interfaces:

- A USB interface to program via computer,
- A µSD card socket for saving images and videos,
- A SPI bus to stream data to LCD shield,
- I2C, CAN and Asynchronous serial bus for interfacing with other sensors and communication modules,
- A 12-bit ADC and a 12-bit DAC,
- Three pins for servomotor control,
- Interrupts and PWM on all I/O pins,
- An RGB LED for informing user.

These all features make OpenMV Cam H7 a good option to use image related applications. The camera module has an OV7725 image sensor. This sensor can take 640x480 8-bit Grayscale images or 640x480 16-bit RGB565 images at 60 FPS. Also, the sensor has an adjustable 2.8mm lens and lenses can be changed regarding needs of the project. This camera module offers applications such as frame differencing, color tracking, face detection, person detection, QR code detection, template matching, and so on.

The camera can be programmed by its own IDE (Open MV IDE). It is an integrated development environment for use with OpenMV Cam and uses MicroPython programming language. MicroPython is an implementation of the Python 3 programming language and includes Python libraries. Unlike other cameras, it does not require extra work for configuring I2C and SPI buses between cameras and microcontrollers.

3.1.2 Communication Module

Another part of this project covers the communication between the node and server. As mentioned previously, NB-IoT is selected as communication technology. Avnet Silica NB-IoT Sensor Shield [42] is used as a communication module because it supports NB-IoT and Cat M1 technologies. Key features of this board are:

- Based on Quectel BG96 Module,
- Global Band support for both Cat M1 and NB1,
- Low power consumption,
- Optional GNSS support,
- Additional connectors such as SIM holder, Arduino pinout and Pmod,
- AT Commands support,
- Built in support for PPP/TCP/UDP/SSL/TLS/FTP(S)/HTTP(S), MQTT.

Some of these features are extremely important for IoT solutions. For example, SIM card holder support on board is necessary to reduce the number of modules connected. NB-IoT communication can be established simply by attaching the SIM card to the holder. Also, Arduino pinouts as shown in Figure 9 is crucial and gives the flexibility to establish data transfer between this module and other modules (camera in this project) via UART serial channel.

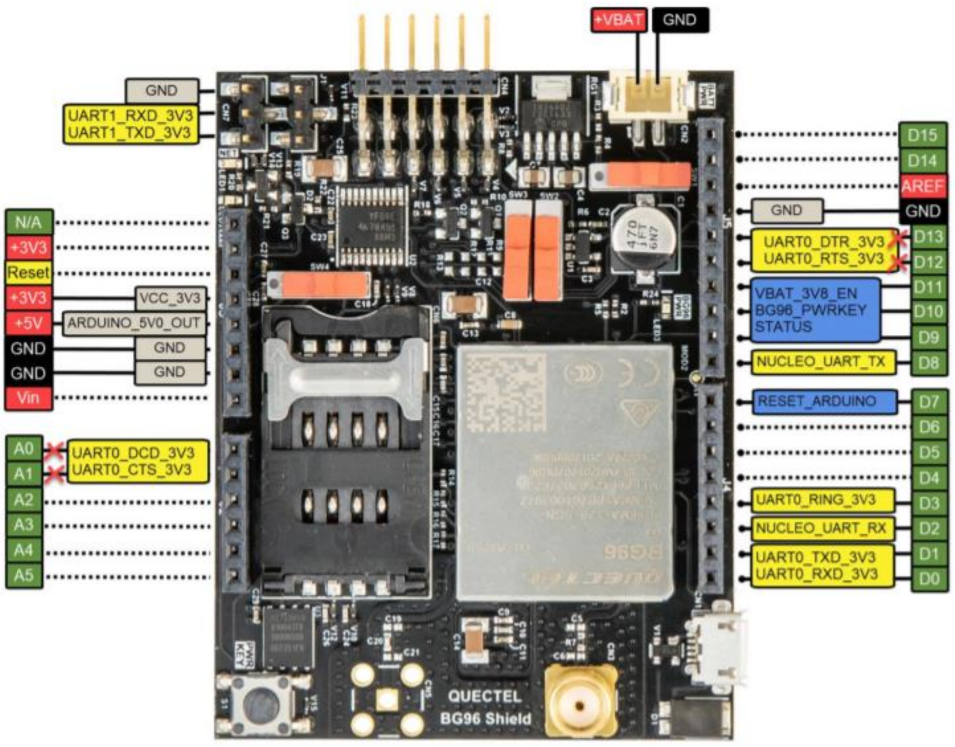


Figure 9. Avnet BG96 Shield pinout [42].

3.2 Data Transfer Protocols

There are several different data transfer protocols supported by the Quectel BG96. These protocols can be utilized with different set of AT commands using Quectel BG96 module. AT commands are instructions used to control a modem. In this section, some of widely used protocols such as HTTP, MQTT, UDP, TCP, and FTP are discussed with their drawbacks and advantages for IoT applications. Also, AT commands required to establish a successful communication for each protocol are explained.

3.2.1 UDP

User Datagram Protocol (UDP) is a transport layer protocol developed by David P. Reed in 1980 and defined in RFC 768 for exchanging messages between network devices [43]. UDP is a lightweight protocol because it does not have long headers meaning it has less signalling overheads. UDP uses divided packets of messages, called datagrams and they are being forwarded by clients (switches, routers and security gateways) to the web server. Packets are sent out to the server without any checks if they received fully or not. Thus, data integrity and reliability are unsecure and out of order. UDP is preferred in applications where data transfer speed is crucial such as livestream videos and online

gaming [44]. In these type of applications, improved connection speed and reduced latency is more important than minor packet loss caused by UDP.
BG96 communication module can create UDP connections with a web server. AT commands shown in Table 3 are written in Quectel BG96 TCP(IP) manual to establish a successful UDP connection [45].

Table 3. AT Commands required to establish TCP connection.

#	AT Command	Explanation
1	AT+QIACT=1	Activate context 1.
2	AT+QIOPEN=1,2, “UDPSERVICE”, “127.0.0.1”,0,3030,0	Start a UDP Service with connection ID 2, context 1
3	AT+QISTATE=0,1	Query if the connection status of context ID is 1.
4	AT+QISEND=2,10,”10.7.89.10”,6969	Send 10 bytes data to remote whose IP is 10.7.89.10 and remote port is 6969.
5	AT+QICLOSE=2	Close the service.

3.2.2 TCP

The Transmission Control Protocol (TCP) is another transport layer protocol which is proposed in 1974 and defined in RFC 675. It is most widely used protocol on the internet because it includes an error controlling system which ensures if packet arrived at the web server or not [46]. However, this property results with a slower connection compared to UDP since it is focus on accuracy rather than speed. Thus, TCP is preferred over UDP for file transfers, webpages and network systems communications. When users open a web browser and enter an URL, TCP is responsible of sending TCP packets from the client (web browser) to the web server (users host address indicated by URL). All packets are sent with a request to bring information or content from the host address. Then, an array of packets is sent to the user by web server. As mentioned in [47], TCP is not an optimal protocol to use in IoT applications because some features are not matching IoT requirements such as, long header size and always-confirmed data delivery which increases the power consumption for IoT devices.
TCP is often compared with UDP, however both protocols are crucial from different aspects. Selection is obviously made depending on the use needs, and they both have some extra features or deficiencies which does not specifically match with massive scale

IoT applications. For example, TCP is a heavyweight protocol because it requires three packets for a socket connection before sending any data. Meanwhile, UDP is lightweight because there is not any packet transmission control, but reliability of data transfer is crucial for some sort of IoT applications such as smart home technologies [48].

BG96 communication module also supports UDP connections. AT commands shown in Table 4 are written in Quectel BG96 TCP(IP) manual to establish a successful TCP connection [45].

Table 4. AT Commands required to establish TCP connection.

#	AT Command	Explanation
1	AT+QICSGP=1,1, “APN”, “”, “”, 0	Configuring a PDP context using network APN.
2	AT+QIACT=1	Activate the PDP context.
3	AT+QIOPEN=1,0, “TCP”, “127.0.0.1”,0,80,0	Open a connection with an IP address 127.0.0.1 and port 80.
4	AT+QISTATE=0,1	Query if the connection status of context ID is 1.
5	AT+QISEND=0,5	Send data using TCP. For example: ➢ Hello Body length is 5 bytes.
6	AT+QICLOSE=2	Close the service.

3.2.3 HTTP(S)

Hyper Text Transfer Protocol (HTTP) is a protocol for transmission of files such as images, videos, audio, or other forms. Early version of HTTP is HTTP/1.1 standardized in 1997 and documented in Request for Comments (RFC) 2068 [49]. Afterwards, HTTP/2 is proposed as a more efficient version and supported in many web servers and browsers in 2015. HTTP is an application layer protocol which runs over TCP/IP protocols, but recently HTTP/3 is proposed with UDP as transport protocol instead of TCP. However, it is not supported by all web browsers at the moment [50]. HTTP and HTTPS are slightly different in terms of security. HTTPS abbreviation is used for HTTP over a Secure Sockets Layer (SSL), where HTTP page requests are encrypted and decrypted by SSL [51].

HTTP protocol consists of set of rules that a server must follow to transfer any data. Requests are sent to server devices by HTTP clients such as web browsers. These requests require methods, and HTTP uses four main methods:

- GET: It is used for requesting data from a web server.
- POST: It is used for sending a data to a web server and creating a resource.
- PUT: It is used for updating a resource on a web server.
- DELETE: It is used for deleting a resource on a web server.

In order to establish a HTTP(s) connection between a server and BG96 communication module, following set of AT commands shown in Table 5 indicated to be used in Quectel BG96 HTTP(S) AT Commands Manual [52].

Table 5. AT Commands required to establish HTTP(s) connection.

#	AT Command	Explanation
1	AT+QHTTPCFG= “contextid”,1	Configure the PDP context ID as 1.
2	AT+QIACT?	Query the state of context.
3	AT+QICSGP=1,1, “APN”, “”, “”,1	Configure PDP context 1. APN is the APN of internet provider.
4	AT+QIACT = 1	Activate context 1.
5	AT+QHTTPURL = 14,80	Set the URL which will be accessed. For example: ➢ www.google.com URL length is 14 bytes.
6	AT+QHTTPPOST=5,80,80	Send HTTP POST request. For example: ➢ Hello Body length is 5 bytes, maximum input and respone time is 80s.

As HTTP runs over TCP protocol, reliability of delivery is high, because TCP protocol consistently exchanges data between client and web server after opening a connection. However, this protocol is mainly designed for web, and works fine between two systems such as a client and a web server only at a time. Whereas, IoT applications include many sensors connected and server may need to respond more than one sensor at the same time. Also, TCP connection should be established first to establish a HTTP connection in both client and server, which results in high energy consumption for IoT device. Additionally, HTTP comes with a biggest message overhead compared to other IoT data transfer protocols, which means higher latency even though with largest bandwidth [53].

3.2.4 MQTT

Message Queuing Telemetry Transport (MQTT) is an application layer communication protocol built on top of the TCP/IP stack, that transports messages between devices. First version of MQTT is proposed by Andy Stanford-Clark (IBM) and Arlen Nipper (Cirrus Link) in 1999 [54] to monitor an oil pipeline in a desert, aiming a bandwidth efficient and low-power consumption protocol. Afterwards in 2013, IBM has announced MQTT v3.1. Then, new version MQTT v3.1.1 is introduced in 2014. Lastly, newest version MQTT v5.0 is standardized in 2019 [55].
MQTT protocol consists of a publish/subscribe architecture to send messages to clients. These messages can be commands to control outputs, published or read data from sensor nodes etc. This is an event-driven architecture where communications are completed over a message broker with several clients such as sensors collecting data. Basically, an MQTT broker is a server receiving all messages from clients and distributes them in an order depending on the application [56]. This order is organized as topics, and these topics are subscribed by clients such as mobile devices, laptops or any other smart device as shown in Figure 10. So, clients do not have any data regarding to subscribers, because broker handles all communication.

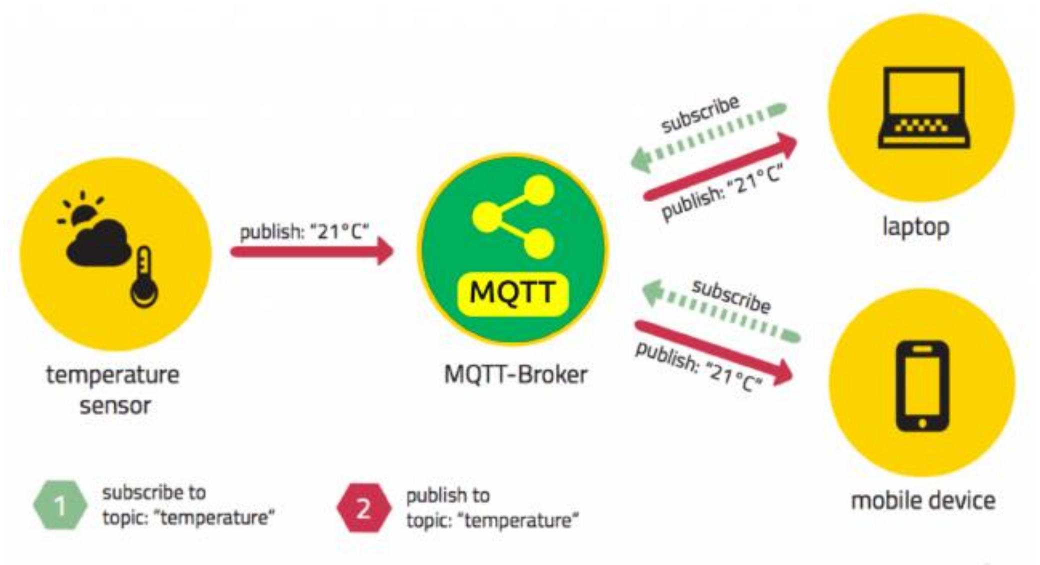


Figure 10. MQTT transmission overview [57].

MQTT is widely compared with HTTP for use in IoT applications. Compared to HTTP, MQTT is a lightweight protocol because HTTP includes many headers and other rules.

Moreover, MQTT allows more devices to connect at the same time compared to HTTP, because in HTTP connection is 1-1 meaning that server responds to one client on a request. Lastly, MQTT is an asynchronous protocol because network and broker itself calculates timing and destination for messages not clients. However, HTTP is a synchronous protocol where clients wait for server response. In overall, MQTT is a lightweight protocol with high flexibility for different IoT devices and services.
Quectel BG96 communication module supports MQTT protocol usage via AT commands. Table 6 indicates the list of commands needs to be used in order to create a MQTT communication with a broker, shared in [58].

Table 6. AT Commands required to establish MQTT connection.

#	AT Command	Explanation
1	AT+QMTCFG= “ali”,0, “0xyz”, “MQTT”, “1xyz”	Configure the device information for Alibaba cloud.
2	AT+QMTOPEN=0, “iot-as-mqtt.cn-shanghai.aliyuncs.com”,1883	Open a network for MQTT client.
3	AT+QMTCONN=?	Connect a client to MQTT server.
4	AT+QMTSUB=?	See topics to subscribe.
5	AT+QMTSUB=0,1, “topic/example”, 2	Subscribe a topic.
6	AT+QMTPUB=0,0,0,0, “topic/pub”	Publish a message. Maximum length supported is 1548 bytes.
7	AT+QMTDISC=0	Disconnect from the MQTT server

3.2.5 FTP

File Transfer Protocol (FTP) is another application layer protocol designed over TCP/IP standard, and it is used for transferring files between clients and servers. It is published in RFC 114 on 1971 by Abhay Bhushan. It is an old protocol still in use today [59], because it is convenient for file transfer as it name stands for it. When a user wants to send files to a specific server, FTP server permits an access to a directory. User can send files to that directory or sub directories using an FTP client. FTP can be used in IoT applications to upload an image or a file consists of different data to a server.
Quectel BG96 communication module also supports file transmission over FTP with AT commands. List of AT commands are given in Table 7 [60].

Table 7. AT Commands required to establish FTP connection.

#	AT Commands	Explanation
1	AT+QICSGP=1,1, “APN”, “”, “”, 1	Configure PDP context, APN is APN of the network
2	AT+QIACT=1	Activate PDP context.
3	AT+QFTPCFG= “contextid”,1	Configure the PDP context ID as 1.
4	AT+QFTPCFG= “account”, “test”, “test”	Set username and password.
5	AT+QFTPCFG= “filetype”,1	Set file type as binary.
6	AT+QFTPOPEN= “ftp.link.url”, 21	Login to FTP server with URL and port number 21.
7	AT+QFTPCWD= “/”	Set current directory.
8	AT+QFTPPUT= “test.txt”, “COM:”, 0	Saving a .txt file to the server with name test.txt.

3.3 Backend Platform

A cloud server is a physical computing unit such as computers, accessed over internet, and used with software and databases run on those servers [61]. This cloud servers are usually located at data centers. Instead of a locally used user device, data storage and computing are completed on servers in data center. This enables users to access applications and files from anywhere in the world. So, cloud servers are used in many aspects of our lives such as email providers like Gmail and Microsoft Outlook or storage providers like Google Drive and Dropbox. Also, cloud technology is one of the major components of IoT because IoT devices collect huge amount of data for different purposes and these all data is sent to a cloud server for further processing and storage. Thus, cloud is crucial for IoT applications allowing devices to connect via communication technologies. Setting up a cloud environment to transfer image is another important part in this thesis, and in this subsection details about backend setup and cloud environment is explained.

Some companies provide IoT clouds for users to connect their devices and transfer data via different protocols such as HTTP, TCP, UDP, MQTT and advanced communication technologies such as LoRa, ZigBee, 3G, 4G and 5G. These IoT cloud services have some advantages over do-it-yourself clouds for IoT enthusiasts. For example, provided platforms usually offer services such as visualization, monitoring, analysis, and simplification [62]. However, users need to develop all these futures by themselves if they want to create their own cloud servers and dashboards. During this thesis, several different cloud server providers and IoT platforms such as Microsoft Azure, ThingsBoard, Thingspeak, and Digital Ocean are tested and used.

Microsoft Azure is a set of cloud services released in 2010 by Microsoft, for organizations and people who wants to use a virtual machine [63] for building, testing and managing applications. A virtual machine is a software computer provided in a physical computer and run applications in an operating system [64]. Microsoft Azure offers students a free account for limited time, where students can create their own cloud servers for projects. During early phases of this thesis, Azure is used as a virtual machine to install ThingsBoard IoT platform. However, after finishing of the student subscription, Microsoft Azure offers its services paid. Therefore, it is not likely to use Microsoft Azure for a student project.

ThingsBoard is an open-source IoT platform for IoT projects [65]. Users can connect their devices using HTTP, MQTT, and CoAP to send and receive data. Then, ThingsBoard offers dashboards, visualizations, analytics and use-case specific features in rule chains for developers as shown in Figure 11. In order to install ThingsBoard, users should select between Community and Professional editions. Professional edition offers extra features and it is paid. During this thesis, Community edition is installed into Microsoft Azure virtual machine with PostgreSQL to control database. ThingsBoard is a strong tool especially for data visualization, however it is not feasible in image transfer because tool does not allow any opportunity to reconstruct the image.

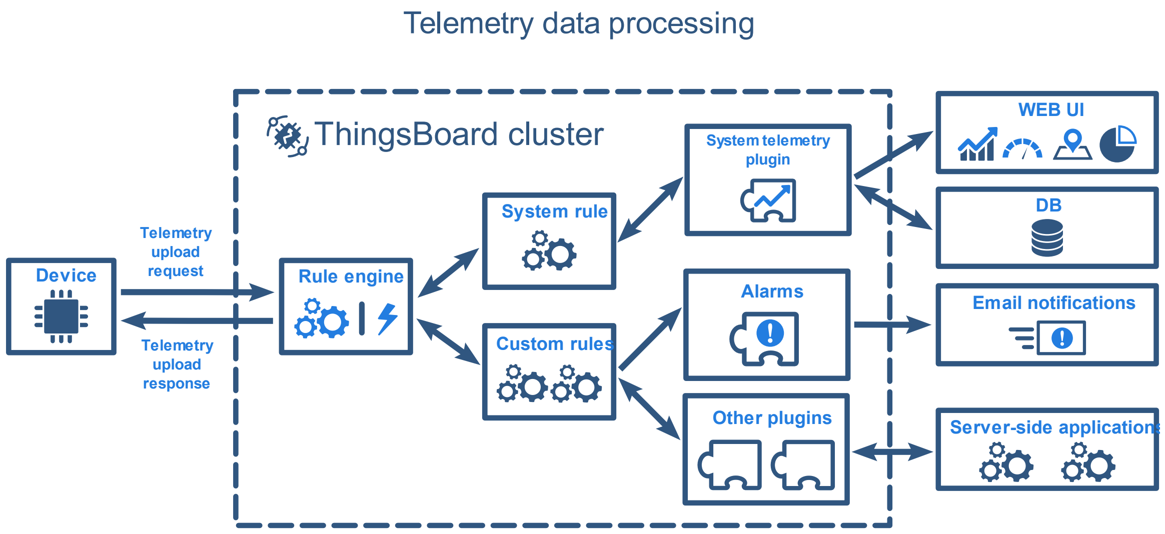


Figure 11. ThingsBoard basic system overview [65].

ThingSpeak is another cloud platform like ThingsBoard which offers real time data collection, analysis and visualization [66]. It is a product by MathWorks, and it allows users to execute MATLAB codes in ThingSpeak for analysis. It supports communication protocols such as HTTP, TCP, MQTT, and UDP. Again, like ThingsBoard, it allows users to visualize sensor data especially in triggering sense. Platform in not practical to transfer an image and reconstruct it. However, ThingSpeak is extremely handy for testing communication between the device and servers. It is used as a proof tool during this project by sending data to the ThingSpeak servers and seeing that data transmission is successful as shown in Figure 12.

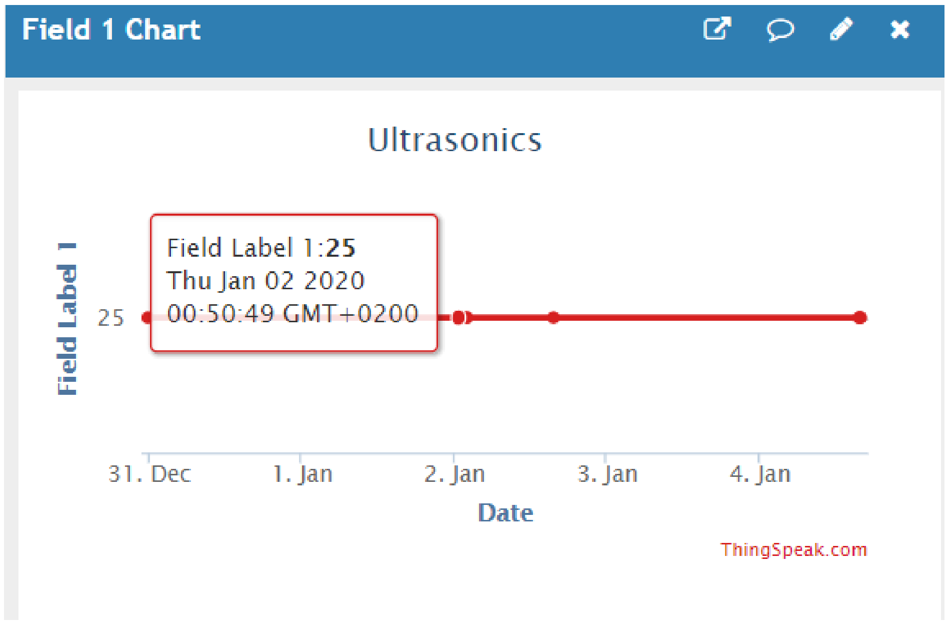


Figure 12. ThingSpeak successful communication.

Digital Ocean is another cloud service provider and internet hosting service founded in June 2011. It allows developers to control their own virtual machines in a user-friendly control panel. Also, it provides different database engines such as PostgreSQL and MySQL to users. In this thesis, Digital Ocean is used for creating an FTP server and sending the image taken by camera to the server using FTP protocol for further processing.

4 Algorithm

Digital image processing is a type of signal processing method to analyse or perform various operations on an image using computers [67]. Useful information can be extracted from an image for different applications from images which are defined in two or more dimensions. Thus, all digital image processing applications can be modelled in multidimensional systems. It is one of the most important technologies among computer science and engineering disciplines, and includes three main steps:

- Acquisition of image using different tools such as camera,
- Performing operations on image,
- Returning extracted features or manipulated images as output.

As mentioned previously in this thesis, camera is another sensor used in smart waste management projects to measure waste level in trash bin. In this chapter, a new algorithm to measure trash level with camera is proposed and explained. Then, it is simulated, and results are discussed.

4.1 Algorithms for Digital Image Processing

In this section, two different image processing algorithms proposed in different works for trash level measurement are analysed. Then, a new algorithm used in this thesis is introduced and explained in detail.

4.1.1 Gray Level Co-Occurrence Matrix

In [8] authors propose a level measurement model based on gray level co-occurrence matrix feature extraction method. Research also focuses on determining best parameter values for GLCM features such as displacement, d, and quantization, G. Moreover, features related to bin level measurement (MLP) are used as inputs to the multi-layer perception and the K-nearest neighbour (KNN) classifiers. Image database is created with bin images from different levels and are used in training and testing with the MLP and KNN classifiers.

GLCM method is a way to extract second order statistical texture features such as homogeneity, entropy, contrast, dissimilarity, correlation, cluster shade, cluster prominence, maximum probability, and energy [8]. These texture features are used to reflect the degree of correlation between pairs of pixels depending on the feature. Before calculating texture features to apply GLCM statistics, probability measure needs to be defined as:

$$P_r(x) = C_{ij}(d, \theta)$$

Therefore, the co-occurrence probability (C_{ij}) between gray levels i and j is defined as:

$$C_{ij} = \frac{P_{ij}}{\sum_{i,j=1}^{G} P_{ij}}$$

Where P_{ij} represents the number of occurrences of gray levels between i and j, withing d, θ, and G. Also, mean (μ) and standard deviation (σ) for the rows and columns of the matrix can be defined as:

$$\mu_i = \sum i \, C_{ij}$$

$$\mu_j = \sum j \, C_{ij}$$

$$\sigma_i^2 = \sum C_{ij}(1 - \mu_i)^2$$

$$\sigma_j^2 = \sum C_{ij}\left(1 - \mu_j\right)^2$$

Using these equations, following texture features can be calculated as:

(1) Energy: $EN = \sum C_{ij}^2$

(2) Entropy: $ENT = \sum C_{ij} \log C_{ij}^2$

(3) Contrast: $CON = \sum C_{ij}^2 \, (i - j)^2$

(4) Dissimilarity: $DISS = \sum C_{ij}^2 \, (i - j)$

(5) Correlation: $COR = \sum \frac{(1-\mu_i)(1-\mu_j)C_{ij}}{\sigma_i \sigma_j}$

(6) Homogeneity: $HOM = \sum \frac{1}{1+(i-j)^2} C_{ij}$

(7) Cluster Shade: $ClusterShade = \sum_i \sum_j \left(i + j - \mu_j - \mu_j\right)^3$

(8) Cluster Prominence: $ClusterProminence = \sum_i \sum_j \left(i + j - \mu_j - \mu_j\right)^4 P_{ij}$

(9) Maximum Probability: $Max\ Prob = MAX_{ij} P_{ij}$

Authors have investigated all co-occurrence texture features related to the parameters d, θ, and G for this specific application. Quantization (G) parameter is crucial for computation of GLCM, and it increases classification accuracy. Displacement (d) varies from 1 to 64, and for large d values classification result reduces. Lastly, orientation (θ)

parameter is considered as less important from all, mentioned in [8]. After testing all parameters and texture features of GLCM, using MLP and KNN classifiers, the authors achieved the best classification results using $d = 1$ and maximum G values, for displacement and quantization. Moreover, best classification from KNN confirmed that KNN=3 and $d = 1$ is best for displacement. Analysing with different databases, best result for bin level measurement is achieved with contrast, entropy, correlation, and homogeneity features. However, designed image processing solution is proposed for cameras mounted on waste trucks, which means trash bins are not monitored all the time. Camera takes two photos, before trash is collected and after, to be used in waste estimation. Also, algorithm is not tested when camera is inside of trash bin because it is mounted on the top of waste trucks.

4.1.2 Hough Transform

In [68] authors deal with the solid waste image detection and classification in waste bin using Hough transform technique for feature extraction, and feedforward neural network (FFNN) model to classify trash level based on learning concept. Various images from trash bin in different status such as empty, medium, full and overflow, are used as input samples for classification. A receiver operating characteristic (ROC) graph is build using results from the rules decision and FFNN.

Hough Transform, patented by Paul Hough in 1962, and subsequently developed by Richard Duda and Peter Hart, is a whole set of algorithms used in computer vision and image processing, making it easy to identify lines and circles in images. It is widely used for detecting edges of a specific figure in an image, transforming between the Cartesian space and a parameter space in a straight line. All the straight lines through a point (x_i, y_i) should satisfy the following Equation 1 with slope m and intercept c in an image. [69]

$$y_i = mx_i + c$$

One can see, each line in an image is expressed by different pair of m and c. So, variables can be reversed and values (m, c) can be rewritten as a function of the image point coordinates (x_i, y_i). Therefore, Equation 1 becomes:

$$c = y_i - mx_i$$

Each point (x_i, y_i) is represented by a line in a (m, c) space in Equation 2. Considering two different pixels P1 and P2 in an image, which intercepts the same line in the (x, y) space, every other possible line intercepting P1 and P2 pixels, will be expressed by a single line in the (m, c) space. There are 5 different algorithmic steps for detecting lines in images coming from this idea:

1. Quantize the parameter space (m, c) to a two-dimensional matrix H with suitable quantization levels
2. Initialize the matrix H to zero.
3. Extract image edges.
4. For each element of H matrix, H(m_i, c_i), increment by 1, which correspond to an edge point. Resulted as a histogram or a vote matrix.
5. Threshold the histogram and take only large valued elements. These elements are lines in the original image.

In [68], authors applied this algorithm for trash bin level measurement, using gradient information to find lines in each frame. Edge lines with high intensity are used to detect presence of trash in waste bin. Using sum squared error function with trained images and features extracted from Hough Transform technique, system reaches to accuracy %82.93. However, amount of the lines in trash bin assumed to express amount of trash in waste bin, but different type of trash may not include a strong linear edge gradient. Additionally, images are taken with a camera from far away, not inside of trash bin as shown in Figure 13. This solution is not applicable when camera is inside of trash bin.

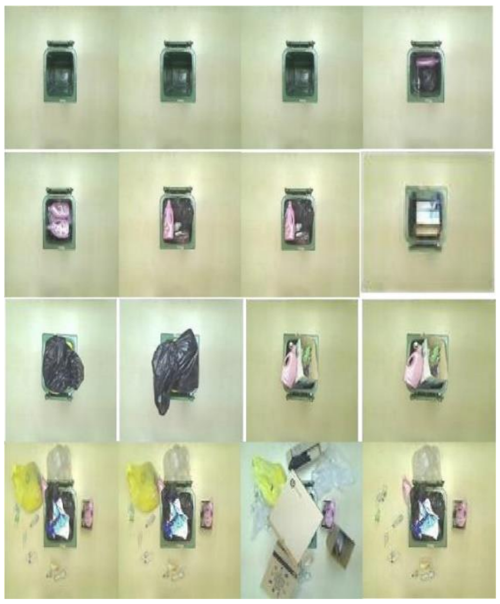


Figure 13. Image data set used in [68].

4.1.3 Grayscale Level Measurement Algorithm

This subsection describes the algorithm proposed to measure waste level in a trash bin in this thesis. Grayscale image of the trash bin is used for this algorithm. A grayscale image represents each pixel in 8-bit, and each pixel has its own value between 0 and 255 by using 8-bit representation and depending on amount of the light. There is none light if pixel intensity value is 0 which means black colour, and there is absolute light if pixel intensity value is 255 which means white colour as shown in Figure 14.

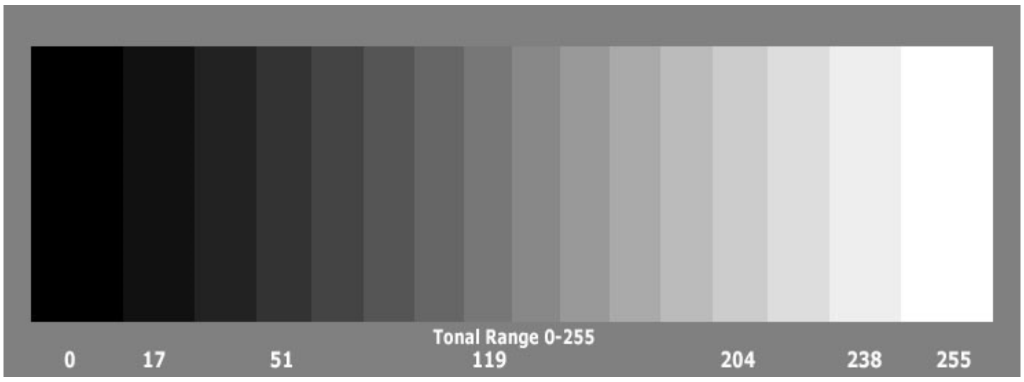


Figure 14. Grayscale pixel intensity values from black to white [70].

It is possible to calculate amount of the waste using a grayscale image if trash bin includes a black bag as shown in Figure 15. Since the background of image is black, new objects in different colour are brighter in a grayscale image. Also, this grayscale image is saved as 2-D matrix of pixel values, therefore these pixel values can be compared with a **threshold** representing the colour of the black bag. As the trash bin starts to fill up as shown in Figure 16, the amount of the white pixels will increase. This increment can be towards four different directions starting from the centre as shown in Figure 16(b) depending on the distribution of waste in trash bin. For example, number of black pixels in y_2 direction is less than y_1 direction in Figure 16(b). y_2 line (or vector in computer program) is full of white pixels, meanwhile y_1 direction has all black pixels. Using pixel values, **fullness percentage** of each line can be calculated as image height and width is known. However, controlling only one side is not enough to decide whether the trash bin is full or not. At least one or more directions should also be compared with the threshold, and this number is referred as **number of controlled sides parameter** in this thesis.

Figure 15. Trash bin used for simulations.

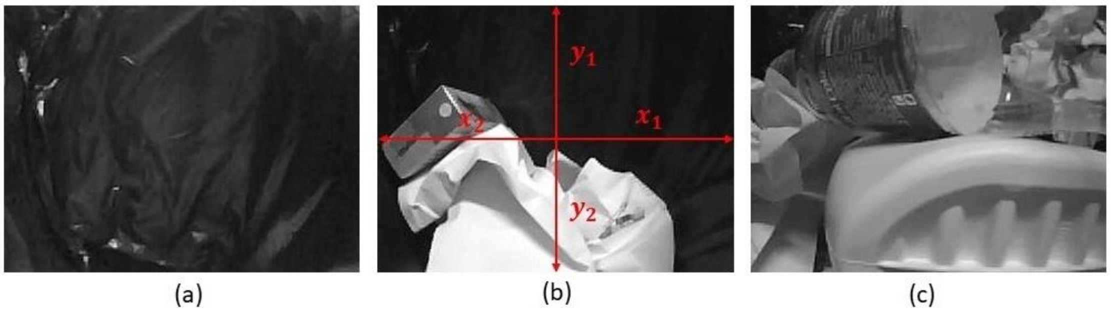


Figure 16. Pictures of trash bin in different states: (a) Empty trash bin, (b) Half empty trash bin with axis coordinates, (c) Full trash bin.

As mentioned previously, a grayscale image is saved as 2-D matrix, and pixel values on x_1, x_2, y_1 and y_2 lines can be compared with a threshold parameter starting from the center to calculate fullness of each vector. However, images consist of many pixels depending on camera resolution, and only controlling one single array from center is trivial to measure level of trash. Therefore, 3 individual arrays from the centre to one single direction are taken as shown in Figure 17 with red and blue lines in image. Then, fullness percentage is calculated for each one, and average of 3 different percentages are taken as a decisive result for that specific side, and this process is repeated for all 4 sides.

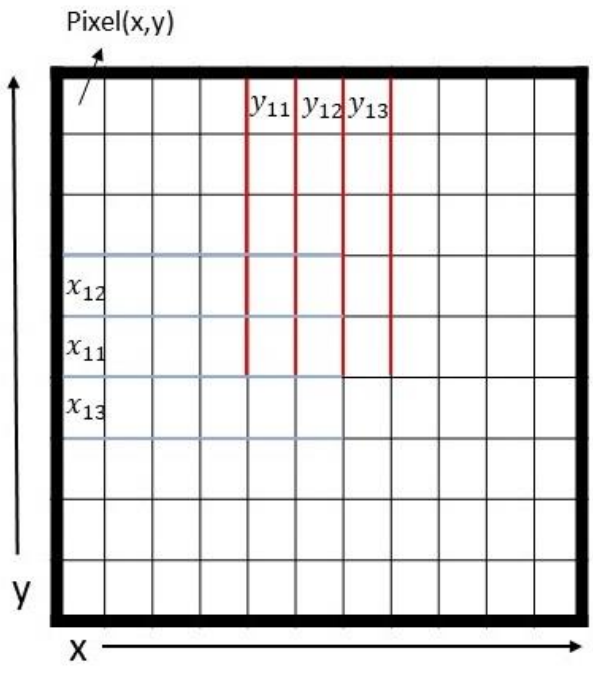


Figure 17. Image pixels shown in 2D matrix with three center arrays for both x and y axis.

When all pixel values for all sides are compared with threshold, and fullness percentages are calculated for each side, overall decision about status of trash bin is made using number of controlled sides parameter as shown in Algorithm I. One can see there are two different for loops for calculating fullness percentages of sides. One of these loops is backwards, because it simply goes from center to top column array ($y_1$in Figure 16) or center to right row array (x_1). Other for loop is normally increasing from center to bottom array (y_2) or center to left row array (x_2).

```
Algorithm I
Initialization:
1: Read image file
2: Set threshold = th;
3: Set fullness percentage = fp;
4: Extract 3 columns from center to top of the image
5: Extract 3 columns from center to bottom of the image
6: Extract 3 rows from center to right-side of the image
7: Extract 3 rows from center to left-side of the image
Start:
8: for i = length of all top column side and all right row side:-1:1 do
9:          if array(i) > th
10:              fullness index = i;
11:         end
12:         fp = ((fullness index - length of array) / length of top array)*-100
13: end
14: for j = 1:length of all bottom column side and all left row side do
```

```
15:         if array(j) > th
16:              fullness index = j;
17:         end
18:         fp  = 100*fullness index / length of array;
19: end
20: for all sides do
21:         average fp = (fp1 + fp2 + fp3) / number of arrays;
22: end
Decision:
23: if sum(average fp (for all four sides) > fp) >= number of
controlled sides parameter
24:         Thrash bin is full.
25: else
26:         Thrash bin is empty.
27: end
```

The proposed algorithm is tested in trash bins which include black trash bags for disposal. These bags are used especially in houses, offices, schools, and other public areas. However, this algorithm has a drawback for a specific situation caused by black and dark grey coloured items. As the algorithm compares the pixel values of black trash bag with other items in the trash bin, black and dark grey items overlap with the trash bag in pixel values. Of course, this drawback highly depends on pixel intensity values of black and dark grey items. For example, some transparent items such as plastic water bottles effects the algorithm decision.

4.2 Simulation Results

As this thesis proposes a new image processing algorithm for trash level detection in trash bins, performance of the algorithm is tested, and results are presented in this chapter. Before simulating the proposed algorithm, two different data sets are created by taking images, and this algorithm is simulated using two data sets. In this chapter, simulation setup and creation of databases, accuracy comparisons with different parameters are documented.

4.2.1 Simulation Setup

In order to start simulations, a trash bin is selected, and a black trash bag is placed in it. Afterwards, the camera is mounted in two different positions as shown in Figure 18 and Figure 19. Then, different amount trashes are put into the bin, consist of white papers, different sized bottles, and some household product bottles. For each picture, amount of the trash and positions of the trash are changed, therefore a diverse data set has been created.

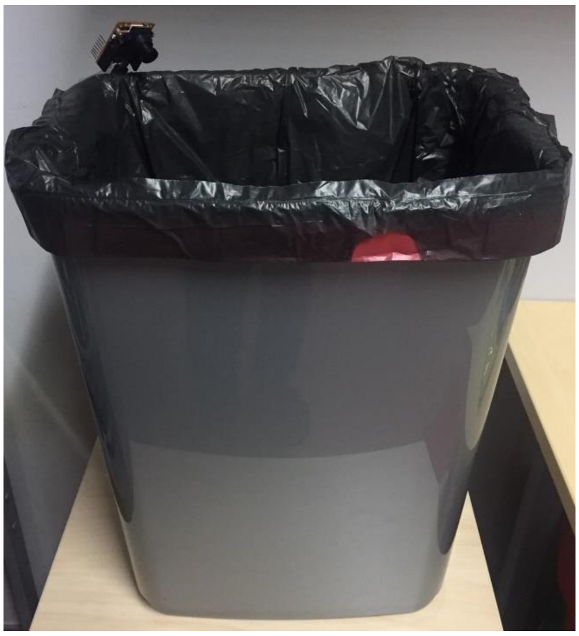


Figure 18. Trash bin with camera placed on the corner side.

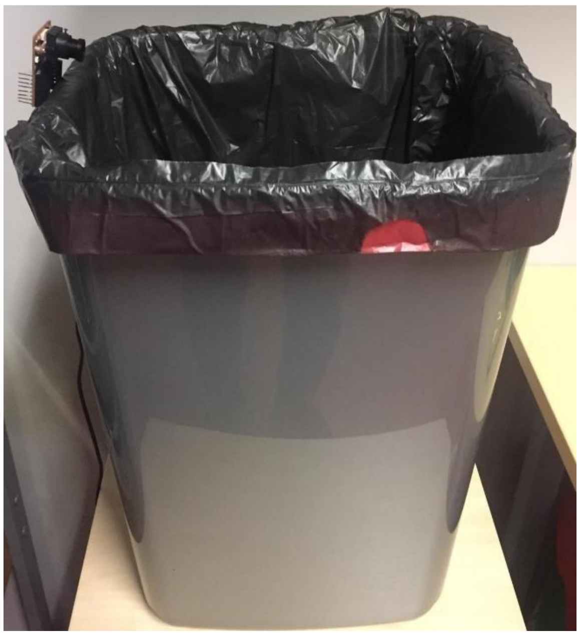


Figure 19. Trash bin with camera placed on the short side.

In order to take pictures and create a data set, snapshot.py script is written. Program imports *sensor, image,* and *pyb* modules as shown in Script I. Then, initializes the camera sensor. Pictures are taken in grayscale and QVGA format to reduce size of the data being transferred. QVGA format is a term used to describe 320x240 resolution, and it is commonly used in smart phones, digital cameras and other LCD displays [71]. Even though higher resolutions are supported by the camera, it is not needed for this purpose because the algorithm depends on amount of the black and white pixels.

Script I: snapshot.py

```
        # import libraries
        import sensor, image, pyb

        # LED pins
        RED_LED_PIN = 1
        BLUE_LED_PIN = 3

        # Initialize the camera sensor.
        sensor.reset()

        # Set pixel format
        sensor.set_pixformat(sensor.GRAYSCALE)

        # Set frame size
        sensor.set_framesize(sensor.QVGA)

        # Let new settings take affect.
        sensor.skip_frames(time = 2000)

        # Red LED is ON for debugging
        pyb.LED(RED_LED_PIN).on()

        # Give the user time to get ready.
        sensor.skip_frames(time = 5000)

        # Red LED is OFF for debugging
        pyb.LED(RED_LED_PIN).off()

        # Blue LED is ON for debugging
        pyb.LED(BLUE_LED_PIN).on()

        # Print console message
        print("Camera is taking snapshot")

        # Take a snapshot and save
        sensor.snapshot().save("example.jpg")

        # Blue LED is OFF for debugging
        pyb.LED(BLUE_LED_PIN).off()

        # Print message
        print("Done! Reset the camera to save the image.")
```

In total 20 different pictures are taken and used for simulation in two different data sets. First data set consists of pictures taken when the camera is placed into short side of the trash bin as shown in Figure 19, and data set is shown in Figure 20. Pictures in the second data set are taken when the camera is placed into corner of the trash bin as shown in Figure 18, and data set is shown in Figure 21. Pictures are numbered from 1 to 10 in both data sets, and trash level in all pictures are given in the Table 8 to calculate accuracy of the program.

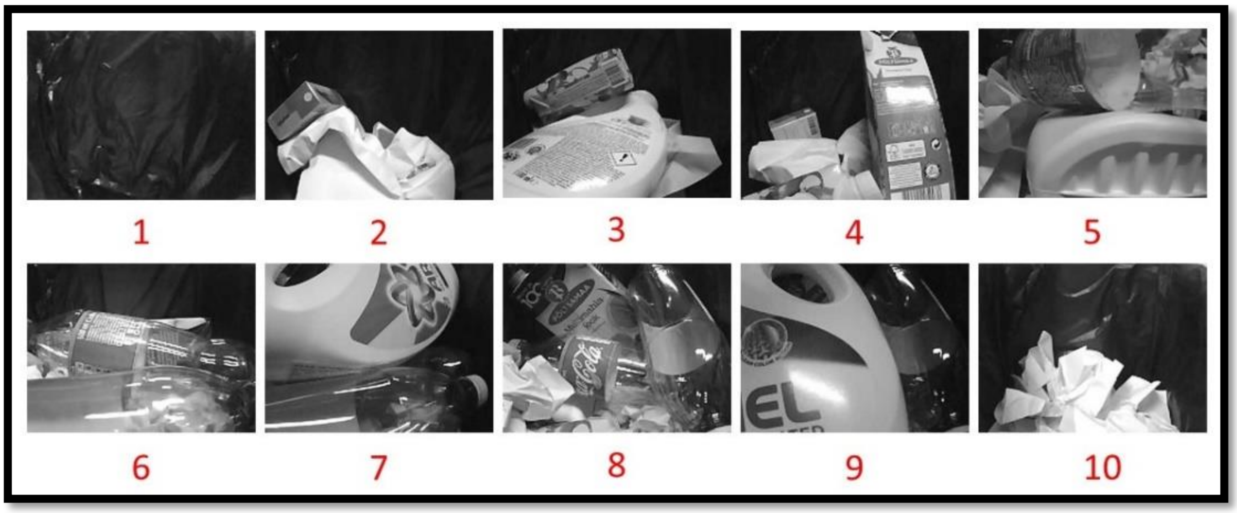


Figure 20. Data set of images taken from the short side angle of the trash bin.

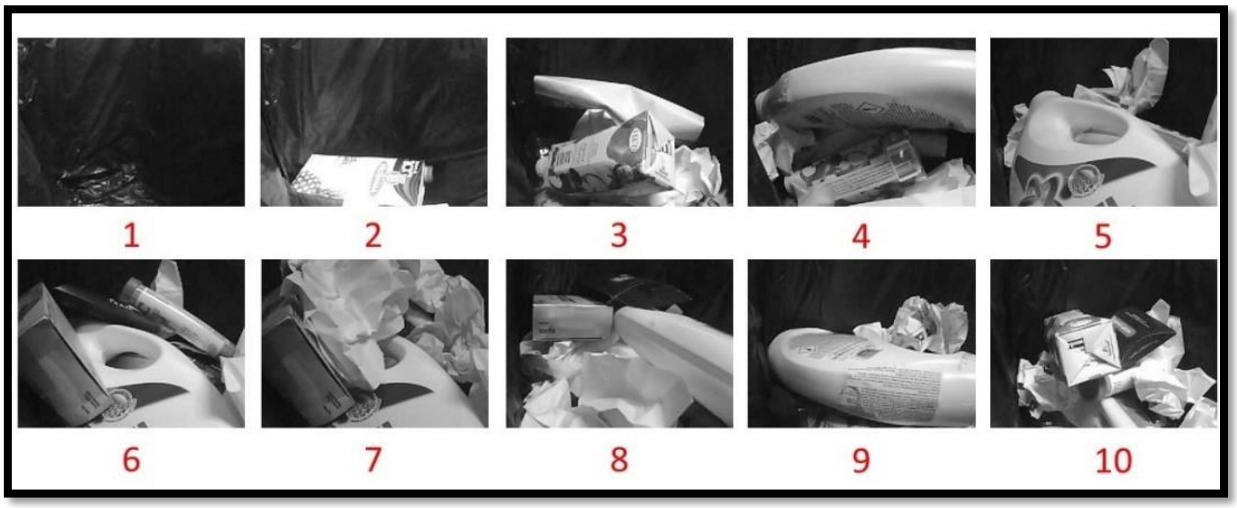


Figure 21. Data set of images taken from the corner side angle of the trash bin.

Table 8. Summary of status of pictures in data sets.

	Short Side Angle	Corner Angle
Sample #	Sample Condition	Sample Condition
1	Empty	Empty
2	Empty (Half)	Empty
3	Empty (Half)	Empty (Half)
4	Empty (Half)	Empty (Half)
5	Full	Empty (Half)
6	Full	Full
7	Full	Full
8	Full	Full

9	Full	Empty (Half)
10	Empty (Half)	Empty (Half)

4.2.2 Results

After completing simulation setup and creating two different data sets, performance of the image processing algorithm is evaluated using MATLAB software. As mentioned previously in the Algorithm chapter, there are two different parameters for the image processing algorithm: threshold for pixel value, and number of controlled sides in the image. Threshold parameter is used for comparing pixel values of the background (black trash bag) and waste inside of the trash bin. It is changed between 90 and 150 to understand effect of threshold on the system performance. These numbers are selected according to colour palette shown in Figure 14. Especially lower limit is important to avoid effects of room lights. However, further investigation about these parameters may increase the algorithm performance. Two parameters are changed one-by-one during simulations and results are compared with Table 8 to calculate accuracy of each option, for both of data sets.

Firstly, photos taken from corner of the trash bin as shown in Figure 21 are simulated with threshold parameter 90 and 150, and number of controlled sides parameter 3. Accuracy of the first simulation is %70 percent for both thresholds as shown in Table 9. In this simulation, even though trash bin is half-full in sample number 4 and 5, algorithm outputs as trash bin is full using 90 as threshold, but outputs are correct with threshold 150. In sample number 4 and 5, distribution of the trash is in front of the camera, and camera fails to measure amount of the trash using threshold 90. However, algorithm correctly decides trash bin is empty with threshold 150, because both images include gray patterns and these two decisions are false positives. Meanwhile in sample number 6, camera angle does not give enough information to measure amount of the trash in the trash bin, and the simulation fails for both cases. Furthermore, in sample number 7 and 8, program fails using threshold 150, because it is a high threshold for comparing white and black pixels, and some gray pixel patterns are included in the photo. Specifically, in sample number 8, dark colored object shown in red box in Figure 22 misleads the algorithm.

Table 9. Results for simulations completed with photos taken from corner of the trash bin. Parameters: NOSC: 3, Th: 90 and 150

Sample # \ Threshold	90	150
1	🟩 Empty	🟩 Empty
2	🟩 Empty	🟩 Empty
3	🟩 Empty	🟩 Empty
4	🟥 Full	🟩 Empty
5	🟥 Full	🟩 Empty
6	🟥 Empty	🟥 Empty
7	🟩 Full	🟥 Empty
8	🟩 Full	🟥 Empty
9	🟩 Empty	🟩 Empty
10	🟩 Empty	🟩 Empty
Accuracy	70	70

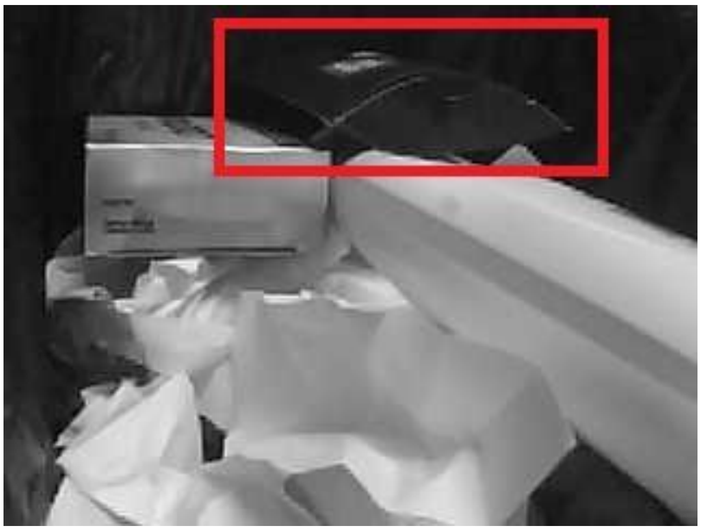


Figure 22. Image of trash bin with red box indicating a black item.

Secondly, same photos from corner side angle shown in Figure 21, are simulated with threshold parameters 90 and 150, but number of controlled-sides parameter is decreased to 2 from 3. Algorithm checked 2 sides in the picture and compared the pixel values with threshold values one-by-one. In this simulation, accuracy of the algorithm for threshold 90 is obtained as %70, and for threshold 150 is obtained as %60, as shown in Table 10. As controlled-sides parameter decreased to 2, algorithm fails in sample number 4, 5 and 9, for both thresholds. Also, in sample number 7, simulation with threshold 150 failed because trash is piled up in front of the camera. However, program does not fail in the same sample using threshold 90 because the black object on the left side of the image misleads the algorithm.

Table 10. Results for simulations completed with photos taken from corner of the trash bin. Parameters: NOSC: 2, Th: 90 and 150

Sample # \ Threshold	90	150
1	Empty	Empty
2	Empty	Empty
3	Empty	Empty
4	Full	Full
5	Full	Full
6	Full	Full
7	Full	Empty
8	Full	Full
9	Full	Full
10	Empty	Empty
Accuracy	70	60

Thirdly, photos taken from the short side of the trash bin shown in Figure 20 are simulated with threshold parameters 90 and 150, and number of controlled-sides parameter 3. As shown in Table 11, program outputs are %80 accurate with threshold 90, and %50 accurate with threshold 150. Results are not accurate especially in tests with 150 threshold value, because sample number 5,6,7,8 and 9 includes transparent bottles. Thus, image has lower pixel values which are close to 0 as shown in Figure 23, and threshold becomes crucial. For example, program does not fail in sample number 7 when threshold is 90.

Table 11. Results for simulations completed with photos taken from short side of the trash bin. Parameters: NOSC: 3, Th: 90 and 150

Sample # \ Threshold	90	150
1	Empty	Empty
2	Empty	Empty
3	Empty	Empty
4	Empty	Empty
5	Full	Empty
6	Empty	Empty
7	Full	Empty

8	Full	Empty
9	Empty	Empty
10	Empty	Empty
Accuracy	80	50

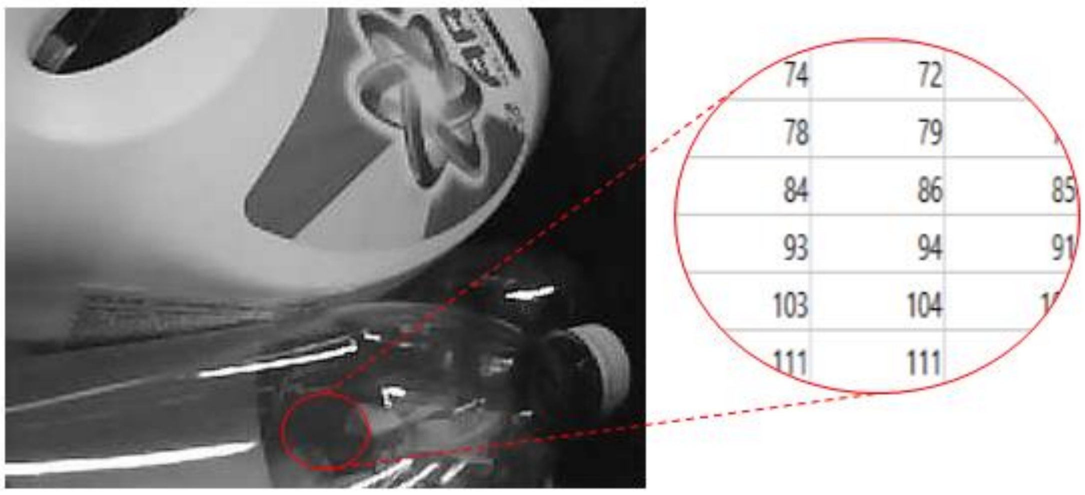


Figure 23. Sample #7 and magnified pixel values as it is saved in program.

Fourthly, photos taken from the short side of the trash bin as shown in Figure 20, are simulated with parameters 90 and 150, and number of controlled-sides parameter 2. As shown in Table 12, program outputs %90 accuracy for threshold 90, and %80 accuracy for threshold 150. Algorithm fails in the second sample with threshold 90, because of the gray-black colored object in the image. When the threshold is increased to 150, algorithm does not fail. However, in sample number 7 and 9, algorithm fails with threshold 150, because of the reason explained previously and shown in Figure 23.

Table 12. Results for simulations completed with photos taken from short side of the trash bin. Parameters: NOSC: 2, Th: 90 and 150

Sample # \ Threshold	90	150
1	Empty	Empty
2	Full	Empty
3	Empty	Empty
4	Empty	Empty
5	Full	Full
6	Full	Full
7	Full	Empty

8	Full	Full
9	Full	Empty
10	Empty	Empty
Accuracy	90	80

After all these simulations, highest accuracy is achieved in fourth simulation using photos taken from the short side of the trash bin, using threshold parameter 90 and controlled sides parameter 2. Simulations shows the threshold parameter is crucial for the image processing algorithm as explained in the section 4.1. After simulating the algorithm using two different thresholds, higher threshold value resulted in lower accuracy because of following reasons:

1. Gray colored objects in the trash bin
2. Transparent items such as plastic bottles

These two reasons are eliminated using a lower threshold value. However, black objects in trash bin cannot be distinguished using the algorithm and those objects can cause false positives.

5 Implementation Considerations

All details about proposed system and possible solutions for smart waste management explained and discussed throughout the previous chapters. Now, this chapter involves all stages of implementation of the platform, and problems faced during implementation in three sections. First section explains how data can be transferred over NB-IoT using different protocols. Second section focuses on the data transmission between camera sensor module and communication module. Last section describes configurations for cloud servers used during this thesis work.

5.1 Data Transfer Over NB-IoT

In Chapter 3, it is mentioned that Avnet BG96 Communication Shield supports different network protocols such as HTTP(s), TCP, UDP, MQTT, FTP(s). Previously, the advantages and disadvantages of all these protocols are explained and differences between them are discussed. Some of these protocols are used during this thesis work for various reasons. In this subsection, all information required to establish a connection and push data is explained with those protocols. Also, there were some problems tackled and they are mentioned in this subsection.

5.1.1 Setup Requirements of Avnet BG96 Shield

Regardless of which protocol is being used, there are some steps that needs to be followed with Avnet Shield. These steps are required to connect the nearest NB-IoT base station before establishing data transmission using any protocol. Quectel QCOM software is used for sending AT commands to the shield. AT commands are instructions used to control a modem [72] .The tool is used for sending and receiving data with serial port. When shield is connected to computer via USB cable, three different ports appear in device manager in Windows:

- AT Port
- DM Port
- NMEA Port

AT Port used with 115200 baud rate for transferring AT commands. After opening the port, PDP context should be activated before data transmission with AT commands shown in Table 13.

Table 13. AT Commands needs to be used before data transmission.

#	AT Command	Explanation
1	ATI	Checking if module is powered or not.
2	AT+COPS?	Checking if module is connected to a network.
3	AT+CREG=2	Registering the module on domain service.
4	AT+QICSGP=1,1, “internet.emt.ee”,””,””,0	Configuring a PDP context using network APN.
5	AT+QIACT=1	Activating the PDP context.

BG96 shield sometimes fails to activate the PDP context, and solution is activating the airplane mode using “AT+CFUN=4” command and deactivating it using “AT+CFUN=1” command. If it does not work, board is restarted using GPIO pin by resetting input power signal. Another important adjustment is required in switches of BG96 shield as shown in Figure 24. Red switches adjust UART Transmission Mode, SIM Card Mode and Power Selection of the shield [73].

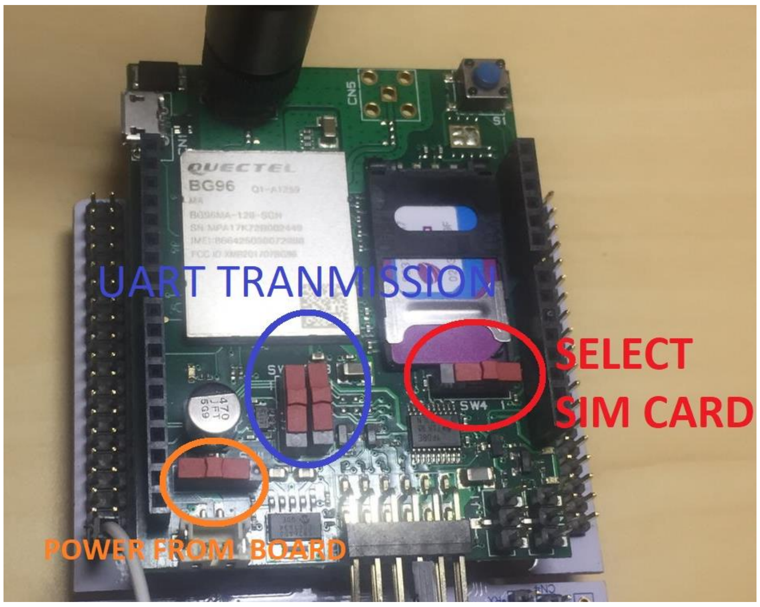


Figure 24. Explanation of Avnet BG96 Shield switches.

5.1.2 Data Transmission with TCP Protocol

TCP protocol is used as a communication check tool in the thesis in some cases. For example, when any other type of protocol fails the transmission, TCP is used to understand if antenna is damaged or defining any other problems regarding to communication. TCP protocol is tested with ThingSpeak servers. After sending AT commands shown in Table 13, another set of commands shown in Table 14 are used in order. In order to send data to specific ThingSpeak server, an API key taken by ThingSpeak website for a specific server must be sent through UART channel with GET method as shown in second step in Table 14. Successful transmission with ThingSpeak server is shown previously in Figure 12.

Table 14. AT Commands used for data transmission with TCP protocol.

#	AT Command	Explanation
1	AT+QIOPEN=1,0, "TCP","184.106.153.149",80	Opening connection with IP address.
2	AT+QISEND=0,46 ➢ GET /update?api_key=yourAPIkey&field1=0	Send data with length 46.
3	AT+QISEND?	Query if data send or not.
4	AT+QICLOSE=0	Close connection.

5.1.3 Data Transmission with HTTP(S) Protocol

HTTP(s) is another protocol tested with BG96 shield and ThingsBoard platform. AT commands written in Table 13 and Table 5 are used in order. However, length of the URL should be adjusted in "AT+QHTTPURL" command. In this case, URL was "http://trashwatch.mooo.com" for ThingsBoard setup, which counts to 26. So, the AT commands becomes "AT+QHTTPURL=26". Then, data is successfully sent using "AT+QHTTPPOST=1000,80,80" command.

5.1.4 Data Transmission with FTP Protocol

Lastly, FTP protocol is used to transfer data to the cloud server. FTP commands required for data transmission was given in Table 7, in Network Protocols chapter. Before sending the file, account details are shared with server to get permission using "AT+QFTPCFG =

“account”, “caner@atakanb.com”, “caner”” command, indicating username and password. Then, login is continued using “AT+QFTOPEN= “ftp.atakanb.com”, 21” command, indicating login URL and port number 21. Lastly, “AT+QFTPPUT= “image.bmp”, “COM:”,0” used to save the image file to the server. For testing purposes, a “.txt” file is successfully uploaded to the server using FTP protocol and QCOM software.

5.2 UART

Avnet BG96 Shield executes AT commands received from UART channel from a USB cable when connected to a computer. UART is a serial communication channel where data format and the transmission speeds are configurable. In a sensor node, this board needs to take commands from the microcontroller unit. As explained in Chapter 3, camera module has its own microcontroller unit with input and output ports. Thus, it is connected to the communication shield via UART channel to transfer AT commands and image. In the main program code, UART library is imported and initialized using three lines of code:

```
from pyb import UART
uart = UART(3,115200,timeout_char=1000)
uart.init(115200,     bits=8,     parity=None,     stop=1,
timeout_char=1000, flow=0)
```

First line imports the library. Second line creates the UART object with 115200 baud rate which is used in BG96 shield. Then, third line initializes the UART bus with given parameters in the method. After initializing the UART in the software, both modules should be physically connected via their serial channel pins regarding to pinout diagrams. UART channels consists of a receiver port (RX) and a transmitter port (TX). When connecting two modules, TX port of camera is connected to RX port of communication module, and RX port of camera is connected to TX part of communication module as shown in Figure 25.

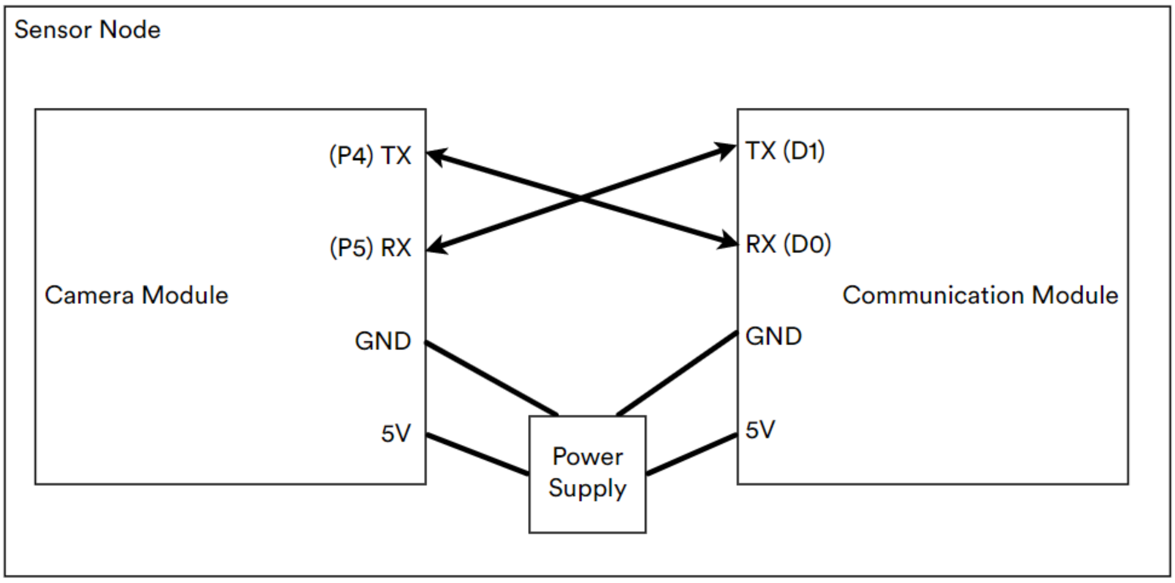


Figure 25. Connections between two modules.

After completing the connections, two different software solutions are implemented for two different network protocols. First of all, AT commands to activate the PDP context and NB-IoT connection mentioned in Table 13 must be sent to the communication module via the UART channel, regardless of the network interface. Then, HTTP(s) is considered to send an image taken by the camera in the early stages of this thesis. Previously mentioned in the Algorithm Chapter, a grayscale image is considered as a 2D matrix in the software. Thus, image pixels are sent to the communication module in pieces through the UART channel from the camera module. For this purpose, two for loops are used to take each pixel in the image and put them into a pixel array and inserted into the HTTP(s) packet to send the communication module as shown in Figure 26. Afterward, the communication module should recognize the AT command for POST request and should send the image pixels to the given server.

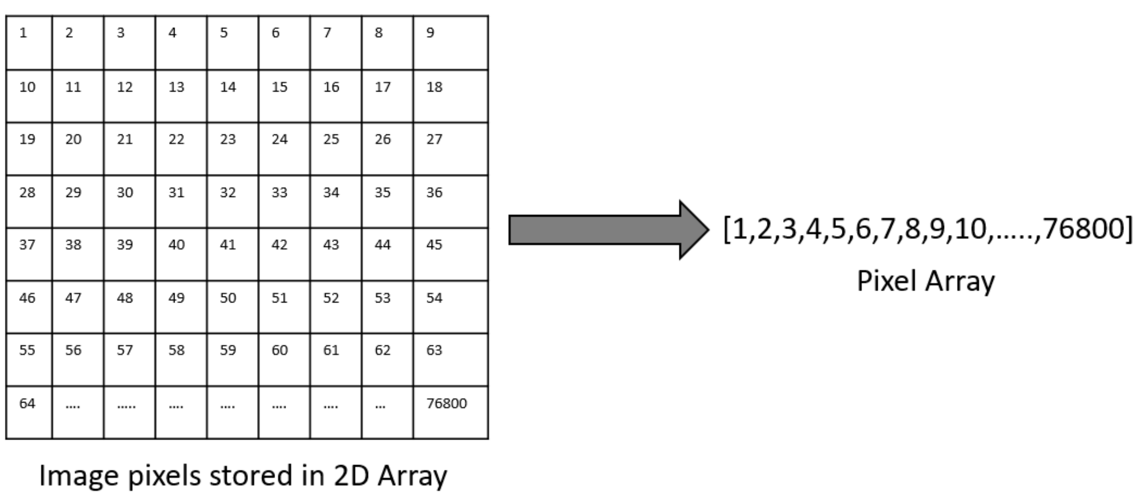


Figure 26. Image shown in 2D array of pixels and transferred as a Pixel Array.

Another software approach used in this thesis work is directly uploading the picture via FTP protocol to an accessed server. In this case, image taken by camera sensor is saved in BMP file format and put into a buffer. Then, this buffer is sent to communication module in string format inside "AT+QFTPPUT" command.

For both solutions, successful data transmission is completed with standalone communication module connected to the computer and using QCOM software. However, transfers have never succeeded with camera connected to the communication module. This problem is due to UART channel transmission between two modules. Different approaches are made to solve this problem. Firstly, port settings such as baud rate, parity, and flow control are checked if they are matching. Secondly, two different software codes are tested to send the taken image to the communication module through the UART channel. Initially, each pixel of the image is sent one by one using a byte array filled with pixels of the image. The second software was aimed to save image as an image object and send it through the channel as shown in Appendix 1, but both software codes have failed. Thirdly, another module (ESP8266) is connected to monitor serial channel using Arduino UNO IDE. However, responses from transmitter of the communication module were not received and monitored. Lastly, a logic analyser is used to monitor serial channel communication, and it is seen that the communication module receives bytes and responses are coming from the transmitter. This shows us serial channel transmits the AT commands and the image, but module does not send it to the cloud server. This problem can be caused by timing of AT commands, because as it is mentioned in BG96 manuals some AT commands require different time delays. It is tested by adding 5ms and 10ms delays between each command in the software, but the problem did not solve.

5.3 Cloud Server Configurations and Final Decision

There are three different cloud servers such as ThingSpeak, ThingsBoard and Digital Ocean are used during this thesis work for different reasons. In this section, server configurations for each server are explained briefly.

ThingSpeak servers are used as a tool for communication tool because it is one of the most basic servers with high ease of use. After registering to ThingSpeak, a channel is created using guide shown in ThingSpeak documentations [74]. Then, write and read API keys and API requests methods are taken from “API Keys” tab to be used in AT commands for data transfer.

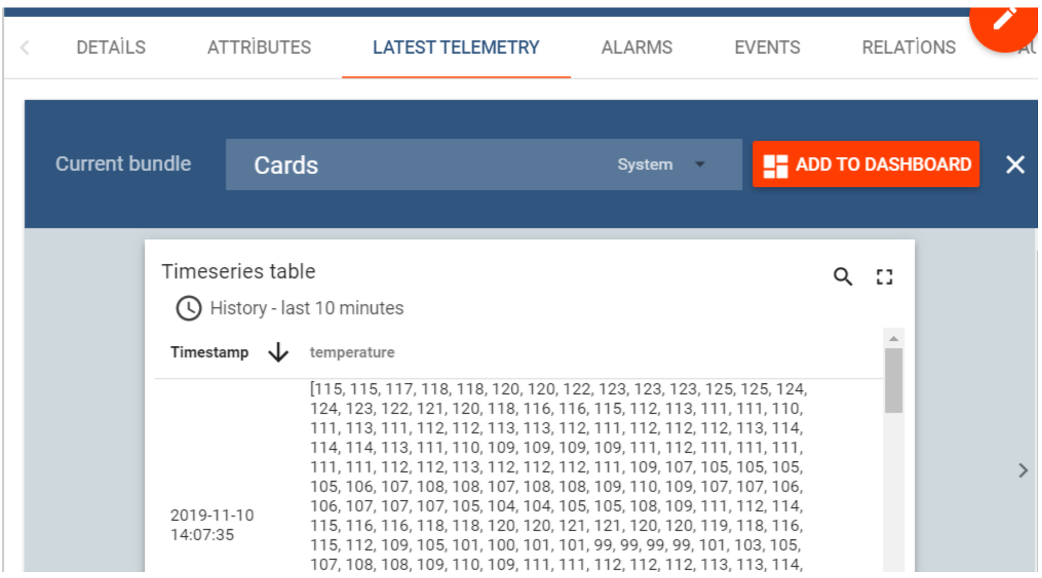
Another cloud server solution was ThingsBoard, used with Microsoft Azure virtual machine. After registration to Microsoft Azure, virtual machine is created through the Azure portal [75]. Operating system is selected as Windows to have flexibility to run Windows-specific software if it is necessary. Also, Windows Server does have a GUI, meanwhile Ubuntu does not. Then, ThingsBoard is installed to the Windows Server in following steps [76]:

- ➢ Installed Java 8,
- ➢ Installed ThingsBoard service,
- ➢ Installed PostgreSQL for database management,
- ➢ Created ThingsBoard database in PostgreSQL.

Moreover, sample cURL command shown below is tested to push data and understand if server operates successfully:

```
curl -v -X POST -d "{\"temperature\": 25}"
$HOST_NAME/api/v1/$ACCESS_TOKEN/telemetry --header "Content-
Type:application/json"
```

HTTP data transmission is tested with BG96 communication shield and QCOM software by creating a new IoT device in ThingsBoard dashboard [77], and an array representing the pixels of an image is successfully sent as shown in Figure 27.



Timestamp	temperature
2019-11-10 14:07:35	[115, 115, 117, 118, 118, 120, 120, 122, 123, 123, 123, 125, 125, 124, 124, 123, 122, 121, 120, 118, 116, 116, 115, 112, 113, 111, 111, 110, 111, 113, 111, 112, 112, 113, 113, 112, 111, 112, 112, 112, 113, 114, 114, 114, 113, 111, 110, 109, 109, 109, 109, 111, 112, 111, 111, 111, 111, 111, 112, 112, 113, 112, 112, 112, 111, 109, 107, 105, 105, 105, 105, 106, 107, 108, 108, 107, 108, 108, 109, 110, 109, 107, 107, 106, 106, 107, 107, 107, 105, 104, 104, 105, 105, 108, 109, 111, 112, 114, 115, 116, 116, 118, 118, 120, 120, 121, 121, 120, 120, 119, 118, 116, 115, 112, 109, 105, 101, 100, 101, 101, 99, 99, 99, 99, 101, 103, 105, 107, 108, 108, 109, 110, 109, 111, 111, 112, 112, 112, 113, 113, 114,

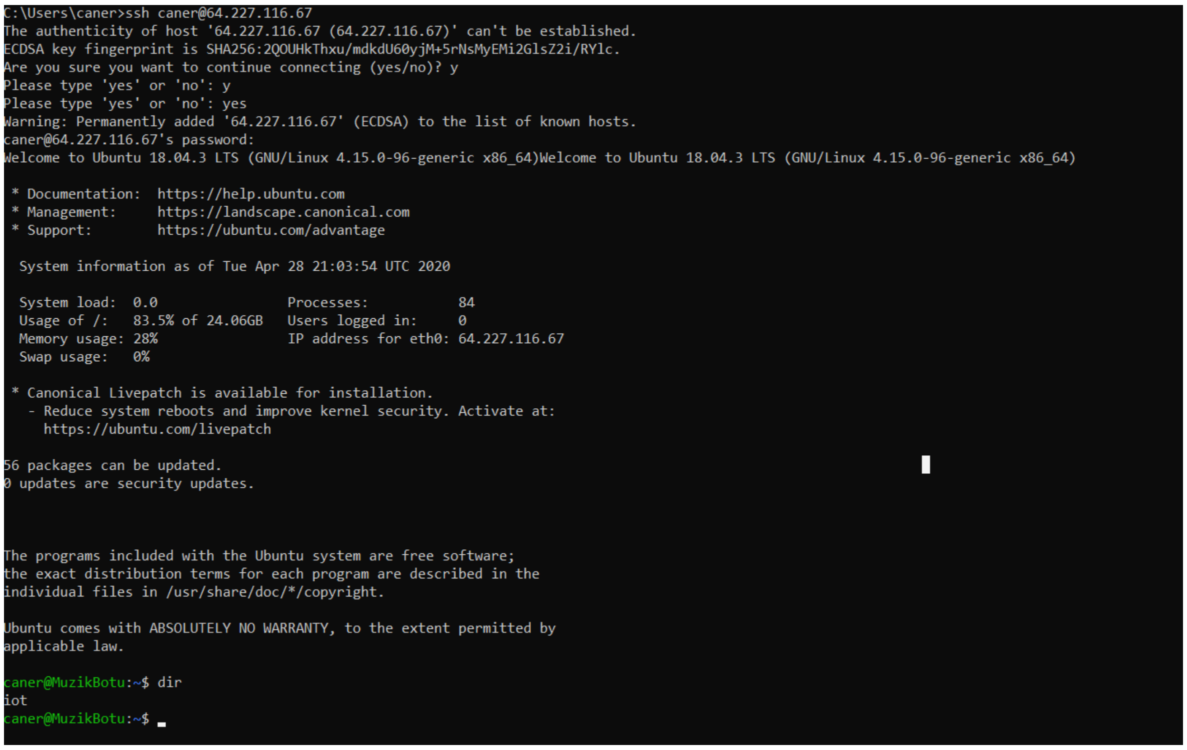
Figure 27. Successful data transmission with ThingsBoard servers.

Lastly, Digital Ocean platform is used for creating FTP server. First of all, according to the Digital Ocean FTP server setup guide [78], an Ubuntu 18.04 server is required as a prerequisite. Thus, Initial Server Setup with Ubuntu 18.04 guide [79] is followed. Basic steps were:

- Logged in as root with server's public IP address.
- Created a new user.
- Granted administrative privileges.
- Set up a basic firewall.
- Enabled external access for user named "caner".

After installing the Ubuntu server, following steps are followed to install "vsftpd" which is a FTP server for Ubuntu:

- Installed vsftpd.
- Opened the firewall.
- Prepared the user directory.
- Configured FTP access.
- Tested FTP access.
- Tested TLS with FileZilla.
- A new folder created named "iot" to save transferred image as shown in Figure 28.



```
C:\Users\caner>ssh caner@64.227.116.67
The authenticity of host '64.227.116.67 (64.227.116.67)' can't be established.
ECDSA key fingerprint is SHA256:2QOUHkThxu/mdkdU60yjM+5rNsMyEMi2GlsZ2i/RYlc.
Are you sure you want to continue connecting (yes/no)? y
Please type 'yes' or 'no': y
Please type 'yes' or 'no': yes
Warning: Permanently added '64.227.116.67' (ECDSA) to the list of known hosts.
caner@64.227.116.67's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-96-generic x86_64)Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-96-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Apr 28 21:03:54 UTC 2020

  System load:  0.0                Processes:           84
  Usage of /:   83.5% of 24.06GB   Users logged in:     0
  Memory usage: 28%                IP address for eth0: 64.227.116.67
  Swap usage:   0%

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

56 packages can be updated.
0 updates are security updates.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

caner@MuzikBotu:~$ dir
iot
caner@MuzikBotu:~$
```

Figure 28. Screenshot of Ubuntu server with “iot” folder.

After configuring the cloud server, the last part we need to take in consideration is how to execute image processing algorithm in it and how to output the final decision. Algorithm is written in MATLAB, and MATLAB supports Linux platforms [80]. MATLAB scripts can be run using following line:

```
    ssh local.foo.com matlab -nodisplay -nojvm < hello.m
```

However, to automatize this process a request is needed. For example, after installing the picture to the FTP server, there should be some sort of API to take the image as an input to the MATLAB script shown in Appendix 2 and take the output for users. This API server can be programmed in Python to create an HTTP post between itself and FTP server and take the image for further processing.

6 Conclusion and Future Work

The rapid growth of the population caused people to live in crowded cities. Therefore, the amount of garbage produced by the people living in the cities is increased, and waste management became an important topic for municipalities. With new developments in technology, now it is possible to monitor numerous trash bins and utilize all data to improve the health of the citizens and optimize costs. This study presented a solution for waste management problem using IoT technology.

In Chapter 2, already existing research papers and IoT systems are analysed and investigated to identify necessary improvements in waste management systems. Thus, it is seen that sensors used for level measurement have important deficiencies such as false alarms caused by uneven loads, and low accuracy rates because of liquid and soft materials. Also, it is found that camera sensors are not preferred in waste management systems due to high costs and lack of image processing algorithms for trash level measurement. Lastly, all communication technologies are briefly examined in Chapter 2. LPWAN technologies such as LoRa, SigFox and NB-IoT stood out with their advantages. These technologies are especially designed for IoT systems. They offer low power consumption, improved ranges and decreased latencies compared to traditional networks. However, NB-IoT is selected as it provides higher data rates and global coverage than LoRa and SigFox.

After selecting the communication technology, an extensive search for hardware modules, network interfaces, and cloud servers have completed in Chapter 3. OpenMV Cam H7 is selected as camera module because it is a low power microcontroller board with a camera sensor. Thus, an additional microcontroller is not used in this project and implementation cost and complexity have been reduced. Then, Avnet BG96 Shield is selected as a communication module which supports NB-IoT. Lastly, network interfaces compatible with NB-IoT such as MQTT, HTTP, TCP, UDP, and FTP are analysed and discussed. FTP protocol is selected to directly upload image as a file to the Backend.

In Chapter 4, a new grayscale-level measurement algorithm for trash bins with black trash bags is proposed. Lack of image processing algorithms on waste management was a huge motive behind proposal of this new algorithm. There are different algorithms as it is also discussed in Chapter 4. However, these algorithms offer trash level measurement methods with a camera placed on waste trucks instead of trash bins, and putting a camera inside of a trash bin and calculating the amount of trash is a completely different scenario because of perspective. Meaning that, observed area is a close-up image of trash bin, and it changes according to where camera is placed. Taking all these into consideration different simulations have completed and, highest accuracy received when camera is placed into short side of the trash bin.

In Chapter 5, implementation considerations for the sensor platform are explained starting from AT commands used to establish data transmission using different protocols over NB-IoT. Then, details about UART serial channel between the communication module and camera module are given, and UART related problems are discussed. Lastly, the cloud server configurations used in this thesis are shown and an FTP server has built on the Ubuntu system using vsftpd, to transmit and save the image for further processing. After transmission of the image, it needs to be an input for the image processing algorithm running on the backend. Then, the final decision about the status of the trash bin can be sent to responsible organizations.

This thesis includes a proposal of an image processing algorithm for specific use, an established communication between the sensor node and a cloud server. It fills the gap of camera sensor-based solutions for smart waste management. However, I can recommend following as future work:

- A graphical user interface is important to transfer final decision to the responsible people. This can be completed with a web or a mobile application.

- Considering the type of trash bin used for simulations, additional experimental tests are required for trash bins placed by municipality because they are bigger.

- Detailed investigations regarding power consumption are required as this platform is an IoT system.

- As mentioned in subsection 5.3, an API server should be designed to take image from FTP server and start image processing steps in the backend.

- Another improvement may be using an adaptive threshold for the algorithm with a wider database to increase detection accuracy of the algorithm.

References

[1] T. Anagnostopoulos, A. Medvedev and K. Kolomvatsos, “Challenges and Opportunities of Waste Management in IoT-enabled Smart Cities: A Survey,” *IEEE Transactions on Sustainable Computing,* vol. 2, pp. 275-289, 2017.

[2] S. Chaudhari and V. Bhole, “Solid Waste Collection as a Service using IoT-Solution for Smart Cities,” *nternational Conference on Smart City and Emerging Technology (ICSCET),* pp. 1-5, 2018.

[3] Statista Research Department, “IoT connected devices worldwide 2030,” Statista, 19 February 2020. [Online]. Available: https://www.statista.com/statistics/802690/worldwide-connected-devices-by-access-technology/. [Accessed 1 May 2020].

[4] K. Mekki, E. Bajic, F. Chaxel and F. Meyer, “A comparative study of LPWAN technologies for large-scale IoT deployment,” *ICT Express,* vol. 5, no. 1, pp. 1-7, 2019.

[5] S. Adarsh, “Performance comparison of Infrared and Ultrasonic sensors for obstacles of different materials in vehicle/robot navigation applications,” in *IOP Conference Series: Materials Science and Engineering,* 2016.

[6] J. Gates, “Fullness Monitoring for Waste: Image-based vs. Ultrasonic Sensors,” Compology, 11 July 2017. [Online]. Available: https://medium.com/@compology/fullness-monitoring-for-waste-image-based-vs-ultrasonic-sensors-29f360bf01e8. [Accessed 2020 May 1].

[7] S. Mahajan, A. Kokane, A. Shewale, M. Shinde and S. Ingale, “Smart Waste Management System using IoT,” *International Journal of Advanced Engineering Research and Science (IJAERS),* vol. 4, no. 4, pp. 2349-6495, 2017.

[8] M. Arebey, “Solid waste bin level detection using gray level co-occurrence matrix feature extraction approach,” *Journal of environmental management 104,* pp. 9-18, 2012.

[9] “GSM (Global System for Mobile communications) Tutorial,” [Online]. Available: https://ecee.colorado.edu/~ecen4242/gsm/index.htm. [Accessed 2020 May 1].

[10] M. Cerchecci, “A low power IoT sensor node architecture for waste management within smart cities context,” *Sensors,* vol. 18, no. 4, 2018.

[11] “What is WiFi: IEEE 802.11,” [Online]. Available: https://www.electronics-notes.com/articles/connectivity/wifi-ieee-802-11/what-is-wifi.php. [Accessed 1 May 2020].

[12] M. A. e. a. Hannan, “A review on technologies and their usage in solid waste monitoring and management systems: Issues and challenges,” *Waste Management 43,* pp. 509-523, 2015.

[13] “Bluetooth Overview,” [Online]. Available: http://www.thewirelessdirectory.com/Bluetooth-Overview/Bluetooth-Overview.htm. [Accessed 1 May 2020].

[14] “What is Zigbee? - Definition from WhatIs.com,” [Online]. Available: https://internetofthingsagenda.techtarget.com/definition/ZigBee. [Accessed 1 May 2020].

[15] “Zigbee explained - What is Zigbee? | Homey,” [Online]. Available: https://homey.app/en-us/wiki/what-is-zigbee/. [Accessed 1 May 2020].

[16] SigFox, “What is Sigfox?,” [Online]. Available: https://build.sigfox.com/sigfox#accessing-the-sigfox-service. [Accessed 1 May 2020].

[17] Link Labs, “What is SigFox?,” 12 February 2015. [Online]. Available: https://www.link-labs.com/blog/what-is-sigfox. [Accessed 01 May 2020].

[18] Semtech, “What is LoRa?,” [Online]. Available: https://www.semtech.com/lora/what-is-lora. [Accessed 01 May 2020].

[19] GSMA, “Narrowband - Internet of Things (NB-IoT),” [Online]. Available: https://www.gsma.com/iot/narrow-band-internet-of-things-nb-iot/. [Accessed 01 May 2020].

[20] Thales Group, “Narrowband IoT,” [Online]. Available: https://www.thalesgroup.com/en/markets/digital-identity-and-security/iot/resources/innovation-technology/nb-iot. [Accessed 1 May 2020].

[21] Y. E. Wang and e. al., “A Primer on 3GPP Narrowband Internet of Things (NB IoT),” *IEEE Communications Magazine,* vol. 55, pp. 117-123, 2017.

[22] R. Ratasuk, N. Mangalvedhe, J. Kaikkonen and M. Robert, “Data Channel Design and Performance for LTE Narrowband IoT,” in *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall),* Montreal, 2016.

[23] u-Blox, “Narrowband IoT (NB-IoT),” [Online]. Available: https://www.u-blox.com/en/narrowband-iot-nb-iot. [Accessed 1 May 2020].

[24] Ericsson, “Network Coverage,” 2019. [Online]. Available: https://www.ericsson.com/en/mobility-report/reports/november-2019/network-coverage. [Accessed 1 May 2020].

[25] Kore Wireless, “LPWAN - Licensed vs. Unlicensed,” [Online]. Available: https://www.korewireless.com/news/lpwan-licensed-vs.-unlicensed . [Accessed 1 May 2020].

[26] S. S. M. a. P. B. G. K. Shyam, “Smart waste management using Internet-of-Things (IoT),” *2nd International Conference on Computing and Communications Technologies (ICCCT),* pp. 199-203, 2017.

[27] K. Nirde, S. M. Prashant and M. C. Uttam, “IoT based solid waste management system for smart city,” in *International Conference on Intelligent Computing and Control Systems (ICICCS),* IEEE, 2017.

[28] G. Prajakta, K. Jadhav and S. Machale, “Smart garbage collection system in residential area,” *IJRET: International Journal of Research in Engineering and Technology,* pp. 122-124, 2015.

[29] F. Folianto, S. L. Yong and L. Y. Wai, “Smartbin: Smart waste management system,” in *IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP),* IEEE, 2015.

[30] S. Zavare and e. al., “Smart City waste management system using GSM,” *Int. J. Comput. Sci. Trends Technol,* pp. 74-78, 2017.

[31] S. S. Navghane, K. M. S. and R. V. M., “IoT based smart garbage and waste collection bin,” *International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE),* pp. 1576-1578, 2016.

[32] S. V. Kumar and e. al., “Smart garbage monitoring and clearance system using internet of things,” in *IEEE International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM),* IEEE, 2017.

[33] S. M. Chaware and e. al., “Smart garbage monitoring system using Internet of Things (IoT),” *Ijireeice 5.1,* pp. 74-77, 2017.

[34] S. S. Ghate and V. K. Sangeeta, "SWACHH: An effective real time solid waste management system for municipality," *International Journal of Computer Applications,* vol. 149, pp. 44-48, 2016.

[35] K. Mahajan and J. S. Chitode, "Waste bin monitoring system using integrated technologies," *International Journal of Innovative Research in Science, Engineering and Technology,* vol. 3, 2014.

[36] M. Al-Jabi and D. Mohammad, "IoT-enabled citizen attractive waste management system," in *2nd International Conference on the Applications of Information Technology in Developing Renewable Energy Processes & Systems (IT-DREPS),* IEEE, 2017.

[37] Raspberry Pi, "Camera Module V2," [Online]. Available: https://www.raspberrypi.org/products/camera-module-v2/. [Accessed 1 May 2020].

[38] Arducam, "A Brief Introduction to Arducam Products," [Online]. Available: https://www.arducam.com/products/. [Accessed 1 May 2020].

[39] ST Microelectronics, "STM32L476RG," [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32l476rg.html. [Accessed 1 May 2020].

[40] I. Abdelkader, "In Search of a Better Serial Camera Module," [Online]. Available: http://sigalrm.blogspot.com/2013/07/in-search-of-better-serial-camera-module.html. [Accessed 1 May 2020].

[41] OpenMV, "OpenMV Cam H7," [Online]. Available: https://openmv.io/collections/products/products/openmv-cam-h7. [Accessed 1 May 2020].

[42] Avnet Silica, "Avnet Silica NB-IoT Sensor Shield," [Online]. Available: https://www.avnet.com/wps/portal/silica/products/new-products/npi/2018/avnet-nb-iot-shield-sensor/. [Accessed 1 May 2020].

[43] Speedcheck, "User Datagram Protocol (UDP)," [Online]. Available: https://www.speedcheck.org/wiki/udp/. [Accessed 1 May 2020].

[44] B. Cole, "UDP and the embedded wireless Internet of Things," [Online]. Available: https://www.embedded.com/udp-and-the-embedded-wireless-internet-of-things/. [Accessed 1 May 2020].

[45] Quectel, "BG96 TCP/IP AT Commands Manual," [Online]. Available: https://www.quectel.com/UploadImage/Downlad/Quectel_BG96_TCP(IP)_AT_Commands_Manual_V1.0.pdf. [Accessed 1 May 2020].

[46] Speedcheck, "Transmission Control Protocol (TCP)," [Online]. Available: https://www.speedcheck.org/wiki/tcp/. [Accessed 1 May 2020].

[47] C. Gomez, J. Crowcroft and e. al., "TCP Usage Guidance in the Internet of Things (IoT)," LWIG Working Group, October 2018. [Online]. Available: https://tools.ietf.org/id/draft-ietf-lwig-tcp-constrained-node-networks-04.html#rfc.section.3.2. [Accessed 1 May 2020].

[48] M. Christian Legare, "Which IoT protocol should I use for my system?," 11 April 2017. [Online]. Available: https://www.embedded-computing.com/embedded-computing-design/which-iot-protocol-should-i-use-for-my-system. [Accessed 1 May 2020].

[49] R. Fielding, J. Gettys, J. Mogul, H. Frystyk and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1," January 1997. [Online]. Available: https://tools.ietf.org/html/rfc2068. [Accessed 1 May 2020].

[50] A. Ghedini and R. Lalkaka, “HTTP/3: the past, the present, and the future,” 26 September 2019. [Online]. Available: https://blog.cloudflare.com/http3-the-past-present-and-future/. [Accessed 1 May 2020].

[51] S. Bhola, “Why HTTP is not suitable for IOT applications,” [Online]. Available: https://www.concurrency.com/blog/june-2019/why-http-is-not-suitable-for-iot-applications. [Accessed 1 May 2020].

[52] Quectel, “BG96 HTTP(S) AT Commands Manual,” 2017. [Online]. Available: http://www.dragino.com/downloads/downloads/NB-IoT/BG96/Quectel_BG96_HTTP%28S%29_AT_Commands_Manual_V1.0.pdf. [Accessed 1 May 2020].

[53] N. Nitin, “Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP,” in *IEEE International Systems Engineering Symposium (ISSE)*, Vienna, 2017.

[54] HiveMQ Team, “Getting Started with MQTT,” [Online]. Available: https://www.hivemq.com/blog/how-to-get-started-with-mqtt/. [Accessed 1 May 2020].

[55] OASIS, “MQTT Version 5.0,” [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html. [Accessed 1 May 2020].

[56] M. Yuan, “Getting to know MQTT,” IBM, 7 January 2020. [Online]. Available: https://developer.ibm.com/articles/iot-mqtt-why-good-for-iot/. [Accessed 1 May 2020].

[57] P. Manzoni, “Intro to MQTT,” [Online]. Available: http://wireless.ictp.it/school_2019/slides/MQTT_v2.pdf. [Accessed 1 May 2020].

[58] Quectel, “BG96 MQTT Application Note,” [Online]. Available: https://sixfab.com/wp-content/uploads/2018/09/Quectel_BG96_MQTT_Application_Note_V1.0.pdf. [Accessed 1 May 2020].

[59] J. Martindale, “What is FTP?,” [Online]. Available: https://www.digitaltrends.com/computing/what-is-ftp-and-how-do-i-use-it/. [Accessed 1 May 2020].

[60] Quectel, “BG96 FTP AT Commands Manual,” [Online]. Available: http://www.dragino.com/downloads/downloads/NB-IoT/BG96/Quectel_BG96_FTP_AT_Commands_Manual_V1.0.pdf. [Accessed 1 May 2020].

[61] Cloudflare, “What Is the Cloud?,” [Online]. Available: https://www.cloudflare.com/learning/cloud/what-is-the-cloud/. [Accessed 1 May 2020].

[62] P. P. Ray, “A survey of IoT cloud platforms,” *Future Computing and Informatics Journal,* vol. 1, no. 1-2, pp. 35-46, 2016.

[63] Microsoft, “What is Azure?,” [Online]. Available: https://azure.microsoft.com/en-us/overview/what-is-azure/. [Accessed 1 May 2020].

[64] vmware, “Virtual Machines,” [Online]. Available: https://www.vmware.com/topics/glossary/content/virtual-machine. [Accessed 1 May 2020].

[65] Thingsboard, “Thingsboard Documentation,” [Online]. Available: https://thingsboard.io/docs/getting-started-guides/what-is-thingsboard/.

[66] ThingSpeak, “About ThingSpeak,” [Online]. Available: https://thingspeak.com/pages/learn_more. [Accessed 1 May 2020].

[67] University of Tartu, “Introduction to image processing,” [Online]. Available: https://sisu.ut.ee/imageprocessing/book/1. [Accessed 1 May 2020].

[68] M. Hannan, W. Zaila, M. Arebey, R. Begum and H. Basri, “Feature extraction using Hough transform for solid waste bin level detection and classification,” *Environmental Monitoring and Assessment 186,* pp. 5381-5391, 2014.

[69] G. Hamarneh, A. Karin and A.-G. Rafeef, “Project Report for the Computer Vision Course Lund: Automatic line detection,” Simon Fraser University, 1999.

[70] N. Joram, “Converting RGB image to the Grayscale image in Java,” 2019. [Online]. Available: https://medium.com/@himnickson/converting-rgb-image-to-the-grayscale-image-in-java-9e1edc5bd6e7. [Accessed 1 May 2020].

[71] V. Beal, “Definition of QVGA,” [Online]. Available: https://www.webopedia.com/TERM/Q/QVGA.html. [Accessed 1 May 2020].

[72] Quectel, “Quectel User Manual,” [Online]. Available: https://www.quectel.com/support/download.htm. [Accessed 1 May 2020].

[73] Avnet Silica, “Nucleo_NbIotBG96_A2_cloud_IBM,” [Online]. Available: https://os.mbed.com/teams/Avnet-Silica/code/Nucleo_NbIotBG96_A2_cloud_IBM/. [Accessed 1 May 2020].

[74] MathWorks, “Collect Data in a New Channel,” [Online]. Available: https://se.mathworks.com/help/thingspeak/collect-data-in-a-new-channel.html. [Accessed 1 May 2020].

[75] Microsoft, “Quickstart: Create a Windows virtual machine in the Azure portal,” [Online]. Available: https://docs.microsoft.com/en-us/azure/virtual-machines/windows/quick-create-portal. [Accessed 1 May 2020].

[76] ThingsBoard, “Installing ThingsBoard on Windows,” [Online]. Available: https://thingsboard.io/docs/user-guide/install/windows/. [Accessed 1 May 2020].

[77] ThingsBoard, “HTTP Device API Reference,” [Online]. Available: https://thingsboard.io/docs/reference/http-api/. [Accessed 1 May 2020].

[78] Digital Ocean, “How To Set Up vsftpd for a User's Directory on Ubuntu 18.04,” [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-set-up-vsftpd-for-a-user-s-directory-on-ubuntu-18-04. [Accessed 1 May 2020].

[79] Digital Ocean, “Initial Server Setup with Ubuntu 18.04,” [Online]. Available: https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-18-04. [Accessed 1 May 2020].

[80] MathWorks, “Start MATLAB on Linux Platforms,” [Online]. Available: https://se.mathworks.com/help/matlab/matlab_env/start-matlab-on-linux-platforms.html. [Accessed 1 May 2020].

Appendix 1 – OpenMV Cam Program Code

```
import pyb, machine, sensor, image, pyb, os, time, _thread
from pyb import UART

# Create and init RTC object. This will allow us to set the current time for
# the RTC and let us set an interrupt to wake up later on.
rtc = pyb.RTC()
newFile = False
uart = UART(3,115200,timeout_char=1000)
try:
   os.stat('time.txt')
except OSError: # If the log file doesn't exist then set the RTC and set newFile
to True
   # datetime format: year, month, day, weekday (Monday=1, Sunday=7),
   # hours (24 hour clock), minutes, seconds, subseconds (counds down from 255
to 0)
   rtc.datetime((2018, 3, 9, 5, 13, 0, 0, 0))
   newFile = True

# Extract the date and time from the RTC object.
dateTime = rtc.datetime()
year = str(dateTime[0])
month = '%02d' % dateTime[1]
day = '%02d' % dateTime[2]
hour = '%02d' % dateTime[4]
minute = '%02d' % dateTime[5]
second = '%02d' % dateTime[6]
subSecond = str(dateTime[7])
newName='I'+year+month+day+hour+minute+second # Image file name based on RTC
BLUE_LED_PIN = 3

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.GRAYSCALE)
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames(time = 1000) # Let new settings take affect.
```

```
pyb.LED(BLUE_LED_PIN).o)
if(newFile): # If log file does not exist then create it.
   with open('time.txt', 'a') as timeFile: # Write text file to keep track of
date, time and image number.
      timeFile.write('Date and time format: year, month, day, hours, minutes,
seconds, subseconds' + '\n')
      timeFile.write(newName + ',' + year + ',' + month +  ',' + day +  ',' +
hour +  ',' + minute +  ',' + second +  ',' + subSecond + '\n')
else:
   with open('time.txt', 'a') as timeFile: # Append to date, time and image
number to text file.
      timeFile.write(newName + ',' + year + ',' + month +  ',' + day +  ',' +
hour +  ',' + minute +  ',' + second +  ',' + subSecond + '\n')
if not "images" in os.listdir(): os.mkdir("images") # Make a temp directory
condition = uart.read()
uart.write("AT+QFTPCFG=\"contextid\",1\r\n")#"AT+QHTTPCFG=\"contextid\",1\r\n
")
print(uart.read())
time.sleep(1000)
uart.write("AT+QIACT?\r\n")
print(uart.read())
time.sleep(1000)
uart.write("AT+QICSGP=1,1,\"internet.emt.ee\",\"\",\"\",0\r\n")
print(uart.read())
time.sleep(1000)
uart.write("AT+QIACT=1\r\n")
print(uart.read())
time.sleep(1000)
uart.write("AT+QIDNSCFG=1,\"8.8.8.8\",\"8.8.4.4\"\r\n")
print(uart.read())
time.sleep(1000)
uart.write("AT+QIDNSCFG=1\r\n")
print(uart.read())
time.sleep(1000)
```

```
uart.write("AT+QFTPCFG=\"filetype\",1 \r\n")
time.sleep(1000)
uart.write("AT+QFTPCFG=\"account\", \"ftpuser\", \"caner0707\"\r\n")
time.sleep(1000)
uart.write("AT+QFTPOPEN=\"trashwatch.mooo.com\",21")
time.sleep(3000)
uart.write("AT+QFTPCWD=\"/iot\"\r\n")
time.sleep(1000)
pyb.LED(BLUE_LED_PIN).off()
img = sensor.snapshot()
uart.write("AT+QFTPPUT=\"araba.txt\", \"COM:\", 0, 11\r\n")
uart.write("Hello World Hello")
bufferSize = img.save('images/' + newName, quality=100).size()
print("Buffer size: " + str(bufferSize))
uart.write("AT+QFTPPUT=\"araba.bmp\", \"COM:\", 0, "+ str(bufferSize) +"\r\n")
time.sleep(1000)
with open('/images/'+newName+'.bmp', 'rb') as imageObject:
    imageByte = imageObject.read(1)
    while imageByte != b"":
        uart.write(str(imageByte))
        imageByte = imageObject.read(1)
print("Done")
```

Appendix 2 – MATLAB Code for Image Processing

```
clc; clear all; close all;
% Taking the picture input
img = imread('C:\Users\caner\Desktop\Tez\Tez Data Set Son\Short Angle\1.jpg');
% Grayscale Pixel Threshold for comparison
th = 150;
% Fullness in percentage
prc = 80;
%% COLUMN PART %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Extracting half columns
halfColTopM = img(1:end/2,length(img)/2);
halfColTopR = img(1:end/2,(length(img)/2)+1);
halfColTopL = img(1:end/2,(length(img)/2)-1);
halfColBotM = img(end/2:end,length(img)/2);
halfColBotR = img(end/2:end,(length(img)/2)+1);
halfColBotL = img(end/2:end,(length(img)/2)-1);
% Calculate fullness percentage of each vector
prc_ctM = funcBackward(halfColTopM, th);
prc_ctL = funcBackward(halfColTopL, th);
prc_ctR = funcBackward(halfColTopR, th);
prc_cbM = funcForward(halfColBotM, th);
prc_cbL = funcForward(halfColBotL, th);
prc_cbR = funcForward(halfColBotR, th);
% Calculate avg. fullness of three vectors
avg_ct = (prc_ctM+prc_ctL+prc_ctR)/3;
avg_cb = (prc_cbM+prc_cbL+prc_cbR)/3;
%% ROW PART %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Extracting half columns
halfRowLeftM = img((length(img(:,1))/2),1:end/2);
halfRowLeftR = img((length(img(:,1))/2)+1,1:end/2);
halfRowLeftL = img((length(img(:,1))/2)-1,1:end/2);
halfRowRightM = img((length(img(:,1))/2),end/2:end);
halfRowRightR = img((length(img(:,1))/2)+1,end/2:end);
halfRowRightL = img((length(img(:,1))/2)-1,end/2:end);
```

```
% Calculate fullness percentage of each vector
prc_rlM = funcBackward(halfRowLeftM,th);
prc_rlR = funcBackward(halfRowLeftR,th);
prc_rlL = funcBackward(halfRowLeftL,th);
prc_rrM = funcForward(halfRowRightM,th);
prc_rrR = funcForward(halfRowRightR,th);
prc_rrL = funcForward(halfRowRightL,th);
% Calculate avg. fullness of three vectors
avg_rl = (prc_rlM+prc_rlL+prc_rlR)/3;
avg_rr = (prc_rrM+prc_rrL+prc_rrR)/3;
if sum([avg_cb avg_ct avg_rl avg_rr]> prc) >= 3
    disp('Trash bin is full')
else
    disp('Trash bin is empty')
end

function [percentage] = funcBackward(halfArray,th)
% Takes top half column and left half row as inputs
% Compares gray pixels with threshold
% Returns percentage of fullness
% Creating a new matrix for max. value calculated in loop
coord = [];
% Compare each element in matrix with threshold
for i = length(halfArray):-1:1
    if halfArray(i) > th
       coord = i;
    end
end
% Calculate how much of the matrix is full (in percentage)
percentage = ((coord-length(halfArray))/length(halfArray))*100*-1;
end
```

```matlab
function [percentage] = funcForward(halfArray,th)
% Takes bot half column and right half-row as inputs
% Compares gray pixels with threshold
% Returns percentage of fullness
% Creating a new matrix for max. value calculated in loop
coord = [];
% Compare each element in matrix with threshold
for i = 1:length(halfArray)
    if halfArray(i)>th
        coord = i;
    end
end
% Calculate how much of the matrix is full (in percentage)
percentage = (100*coord)/length(halfArray);
end
```