

University of Engineering and Technology Peshawar,
Jallozai campus

Acquisition of Data from eBike and Uploading to the Cloud for Extensive Analysis

By

Nizam Ahmad

Fahad Aziz

Farhan Ali

A thesis submitted to the University Engineering and technology Peshawar,
Jallozai campus of the requirement of degree of

Bachelors in computer science and information and technology from
department of CS&IT

Supervisor: Dr. Bilal Habib

The undersigned certify that the supervisor “Dr. Bilal Habib” recommend to the University of Engineering and Technology Peshawar for acceptance of this thesis for the purpose of fulfilling the requirements for the degree stated.

Supervisor: Dr. Bilal Habib

Signature: _____

Date: _____

DEDICATION

To begin, we would like to express our gratitude to "Almighty Allah" for blessing us with guidance, strength, mental power, protection, skill sets, and a long and healthy life.

The moral, spiritual, emotional, and financial support of our dear parents is recognized and appreciated in this study. They have been our motivators and strength when we felt like giving up. To our parents, siblings, close friends, and fellow students who supported us in completing our research.

Additionally, we would like to express our gratitude to our supervisor, Dr. Bilal Habib, for their unconditional support, helpful information, and amazing guidance. He also made time for us whenever we needed it or we ran into issues.

ACKNOWLEDGEMENTS

*Our supervisor, **Dr. Bilal Habib**, made this work possible, so I would like to acknowledge him as well as my sincere appreciation. Our progress with this project was assisted by his direction and guidance at every stage.*

We would also like to thank our teachers for their time and useful comments, which helped us in improving our final year project. We would also like to thank the staff of the University of Engineering and Technology Peshawar, Jallozai Campus for their support and assistance.

Last but not the least, we would like to thank our parents for their unconditional love and support throughout our bachelor's degree.

ABSTRACT

Transportation demand is rising along with intense human activity in various places. Due to its flexibility, a motorcycle may be a better option for only certain dense roads. This could result in massive use of motorcycles with fossil fuel engines. This results in significant air pollution as well as high fuel consumption. Electric bike is expected to be the answer for both problems, since it does not cause air pollution and does not require fossil fuel to operate. However, this bike has range and speed anxiety. It depends on energy stored in the battery. This difficulty makes the good points of the bike unattractive. The project “Design of Data Acquisition System for E-Bike to develop Drive-cycle for Peshawar” is used to develop a system for Electric bike to collect data from the Electric bike to develop a Drive cycle for the better optimization of the battery. For acquiring data acquisition system for E-Bike different sensor are used to take data from the Electric bike. A memory card is used to store this data. In order to collect data from the bike, display it on the LCD, and store it in memory, Arduino interfaces with current, voltage and GPS.

1	CHAPTER 01 INTRODUCTION	1
1.1	Introduction	1
1.2	Motivation	2
1.3	Overview	3
1.4	Background	4
1.5	Research problem	5
1.6	Research objective	5
1.6.1	<i>Designing Data Acquisition System</i>	5
1.7	Applications	6
1.7.1	<i>Range Extension</i>	6
1.7.2	<i>Battery Optimization</i>	6
1.8	Limitations	6
1.9	Thesis Structure	7
2	CHAPTER 02 LITERATURE REVIEW	8
2.1	LITERATURE REVIEW	8
3	CHAPTER 03 METHODOLOGY	10
3.1	Design	10
3.2	Block Diagram	11
3.3	Flow Chart	11
3.5	Implementation	12
3.5.1	<i>Testing</i>	13
3.6	Software & Hardware	14
3.6.1	<i>Software</i>	14
3.7	Hardware	17
	Hardware oscilloscope	30
	Signal generator	31
4	Chapter 04 Collecting Data through Sensors	32
4.1	ACS712 Current Sensor for DC Current	32
4.4	DC Voltage Sensor	33
4.6	GPS	35
4.7	SD Card Module	36
5	Analog to Digital Conversion & SD Card Testing	38
5.1	Sampling	38
5.1.1	<i>Timers and Interrupts</i>	39

<i>Figure 25 Prescaler and the Compare Match Register</i>	39
5.1.2 <i>Code Structure</i>	41
5.1.3 <i>Timer ISR ()</i>	41
5.2 <i>Digitizing Analog Signals</i>	42
5.2.1 <i>Analog to Digital Conversion</i>	43
5.2.2 <i>ADC ISR ()</i>	44
5.2.3 <i>Results</i>	45
5.3 <i>SD Card Testing</i>	45
5.3.1 <i>Examine for Missing Cycles</i>	46
5.3.2 <i>Separate Data Transfer Speed</i>	47
5.3.3 <i>Concatenated Data Transfer Speed</i>	48
5.3.4 <i>Time Taken: Analog to Digital Conversion & Data Transfer to SD Card</i>	49
5.4 <i>Conclusions</i>	50
6 UPLOADING DATA TO CLOUD & SOME OPERATIONS	51
6.1 <i>DESIGN</i>	51
6.2 <i>S3 BUCKETS CREATION</i>	51
6.2.1 <i>Introduction</i>	51
6.2.2 <i>Source Bucket</i>	52
6.2.3 <i>Destination Bucket</i>	52
6.3 <i>LAMBDA FUNCTION</i>	53
6.3.1 <i>Introduction</i>	53
6.3.2 <i>Creating lambda function</i>	54
6.3.3 <i>Attach policy to IAM role</i>	55
6.3.4 <i>Adding a trigger for our Lambda function</i>	56
6.3.5 <i>Adding code to our Lambda function</i>	56
6.3.6 <i>Testing our Lambda function</i>	57
6.3.7 <i>Results</i>	58
7 CONCLUSIONS AND FUTURE WORK	59
7.1 <i>Conclusion</i>	59
7.1.1 <i>Important Finding</i>	59
7.1.2 <i>Future Recommendation</i>	59

LIST OF ABBREVIATIONS

AC	Alternating Current
DC	Direct Current
ADC	Analog-to-Digital Conversion
DAC	Digital-to-Analog Converter
GPS	Global Positioning System
OS	Operating System
RMS	Root Mean Square
OCR	Out Put Compare Register
CTC	Compare Time Count
S3	Simple Storage Service
SD card	Secure Digital Card
ISR	Interrupt Service Routine

LIST OF SYMBOLS

V	Volt
W	Watt
A	Ampere
mm	Millimeter
Hz	Hertz
	Giga
GHz	Hertz

1 CHAPTER 01 INTRODUCTION

1.1 Introduction

The most useful form of energy for various applications, including transportation systems, is electrical energy. This is because energy can be produced easily using a variety of resources and is widely applicable to a wide range of uses. The transportation equipment must perform fundamental changes in order to use electrical energy for the system. A battery stores the energy and a motor powered by electricity replaces the device's primary motor. Electric transportation systems are still in the early stages of development, and more research is still required to determine how to improve the system and how to address some of its shortcomings.

Range and speed anxiety with electric bikes are two of the most significant problems. Its speed and range are limited to a particular area. It is determined by a number of variables, such as the battery size, typical speed, terrain, rider weight, and others.

The battery range of an electric bike is decreased by heavier riders and bikes. Keep in mind that carrying luggage or people will require you to pedal more forcefully or increase your level of assistance, both of which will cause the battery to discharge more quickly. The maximum travel distance would be lowered as the rider's weight increased due to the increased energy demand on the battery supply. Similar to driving a car, frequent stops and starts use up a lot of battery power. It is unavoidable when travelling in a city with frequently changing traffic lights. So, when you want to move forward, instead of turning on the high assistance level, shift into a lower gear and help your bike start. That could reduce your starting speed. However, it will significantly impact battery conservation for your electric bike. Additionally, avoid sudden

acceleration because this causes your bike's battery to release a burst of energy. Your battery's range is reduced as a result of the increased power consumption per time. If your bike's frame is comfortable and you're riding in an upright position, it will be more difficult for you to accelerate and maintain a high speed. The battery will run out of power more quickly, though, if you rely on motor assistance to help you move faster. The type of terrain you are driving on will also influence how frequently you use your brakes, which negates the energy the battery produces.

This project suggests an embedded process for data measurement and acquisition of an electric bike based on the mentioned description. This makes it possible to display some crucial information on the dashboard. Some sensors are needed in the developed instrumentation system for measurement purposes. A processor is tasked with performing these tasks because some data or information may not have been directly displayed from measurements but rather requires some calculations. The processor gathers measured data from the sensors and computes it to produce the required information. The system uses an Arduino Uno as a data acquisition device for data acquisition. On an LCD, information and data from measurement and calculation are displayed. These comprise current, voltage, power, the location of the electric bike, as well as other information based on driving requirements. These facts and figures are essential for safe and efficient driving that results in peak performance.

1.2 Motivation

The goal of this project is to increase the speed and range of an electric bike. In general, electric bikes are slower and less powerful. The distance an electric

bike can travel on a single charge is referred to as range. E-bike range is limited. Even with brand-new batteries, it would be risky to predict a range of more than 30 km when riding in cities. This means that unless you have a chance to charge the batteries for an hour or longer at your place of halt, it would not be advisable to travel more than 15 km from home (since you would need to travel the same distance back). This also implies that you might not have enough battery life to accommodate unforeseen additional travel throughout the day.

The recent e-bike industry trend doesn't seem to be slowing down any time soon. It just seems to be getting bigger over time, which is good. E-bike sales are anticipated to rise in 2022 after five years of constant growth. In many parts of the world, electric bikes are becoming more and more well-liked as an eco-friendly form of transportation. This is primarily because electric bikes have so many advantages over other forms of transportation, including reduced CO2 emissions, air pollution, noise pollution, fuel costs, improved public health, reduced traffic congestion, and reduced costs for constructing and maintaining road infrastructure. To address these issues and encourage more people to use electric bikes for environmental benefits, we created this project.

1.3 Overview

Electric bikes and cars are associated with the *range-anxiety*. It means the rider, or the driver is always anxious about the potential range that can be achieved with the available electric power. We plan to mitigate this problem in our project.

In this project we are proposing to install multiple sensors over the electric bike. These sensors will sense the current, voltage and power consumption of the bike. Moreover, the location, speed and distance covered will be calculated. It will help

in determining the range and potential speed the rider can achieve with the available battery power. We will display the data to the driver and also analyze it to further refine an e-bike product.

We plan to extract the sensors data using microcontrollers. These controllers have built-in ADCs (analog-to-digital convertors). The data extracted will be stored in the memory. Furthermore, we plan to upload it to the cloud. We will do extensive data analytics over the cloud. In future, we also plan to collect data from many e-bikes simultaneously and provide an uber like ride-sharing services.

1.4 Background

E-bikes are typically less powerful and slower. The distance an electric bike can go on a single charge is referred to as range. The range of electric bikes is limited. Expecting a range of more than 30 km while driving in a city would be dangerous, even with brand-new batteries. This means that unless you have the ability to charge your batteries for an hour or more at a stopover, it would not be advisable to travel more than 15 km from home (as you would have to travel the same distance back). Additionally, this implies that you might not have enough charges to cover unforeseen extra travel during the day.

E-bikes typically have less power than traditional bikes and are also slower. The "range" of an electric bike refers to the maximum distance that it can travel on a single charge. There is a limit to the range that electric bikes can travel. Because we do not communicate with the hardware's, we are unable to improve the range of E-Bike's But if we use a few sensors to communicate with it, we will be able to do so. Just because of that, it is not recommended that you travel more than 15 kilometers away from your residence unless you have the ability to charge your batteries for at least an hour while you are stopped somewhere (as you would have to travel the same distance back). In addition, this suggests that you might not have sufficient charges to cover any unexpected additional travel that might occur throughout the day.

1.5 Research problem

Speed and range restrictions are issues with electric bikes. Countless factors impact the electric bike's speed and range. The battery's capacity is one of the factors that affect an electric bike's range. The battery range of an electric bike is lowered by heavier riders. Battery power is heavily drained by frequent stops and starts. It takes much more effort to ride a bike up a steep slope than it does on flat ground. The environment's temperature also has an effect. As a result, the battery has a lower energy capacity at zero degrees, which reduces the amount of distance you can travel. In this project, we introduced a data acquisition system to collect data and do extensive analysis over the cloud in an attempt to solve this issue.

1.6 Research objective

We plan to extract the sensors data using microcontrollers. These controllers have built-in ADCs (analog-to-digital convertors). The data extracted will be stored in the memory. Furthermore, we plan to upload it to the cloud. We will do extensive data analytics over the cloud. In future, we also plan to collect data from many e-bikes simultaneously and provide an uber like ride-sharing services.

1.6.1 Designing Data Acquisition System

The Arduino is interfaced with DC voltage and DC current sensors to measure the battery voltage and the current drawn by the load, respectively. DC voltage and DC current are used to calculate DC power. Similar to this, the Arduino is interfaced with AC voltage and current sensors to determine the AC voltage and current of the load. AC power is determined by multiplying AC voltage by AC current. Using a GPS sensor, the bike can be located. Through a memory module, the entire dataset is stored on the memory card.

1.7 Applications

The project “Design of data acquisition system for Electric bike ” includes the applications listed below.

1.7.1 Range Extension

The range represents the typical distance that a bike can travel. Therefore, range extension basically means to increase the average distance that a bike can travel. One use for this project is to increase the range of an electric bike so that it can be used to travel over long distances.

1.7.2 Battery Optimization

Making the best use of the battery is known as battery optimization. For lengthy travel, it enables a slow battery discharge. Overheating is prevented for the battery.

1.8 Limitations

There are some circuitry-related restrictions on this project. Which are outlined shortly

- The 1602A's small LCD screen cannot display all the information at once.
- Transferring data to an SD card at a slow rate. GPS displays coordinates after a delay.

1.9 Thesis Structure

This project helps in developing Drive Cycle for Peshawar for an e-bike. In particular, it contributes to rigorously testing established algorithms using current sensors, voltage sensors, GPS sensors, and memory modules to introduce new methods in the field and apply new techniques to discover novel objects. The structure of the thesis is as follows: The first chapter discusses the underlying motivations and background of research. Various perspectives of data acquisition system research are dealt with in Chapter 2 of the Literature Review. This chapter also describes the types of existing data acquisition systems. Finally, this chapter outlines the existing software toolset and selects the appropriate algorithmic component for the discovery module. Chapter 3 goes through a detailed discussion on methodology. Chapter 4: Collecting data through sensors, results and a detailed discussion of our designed detection method and its implementation using various hardware components. Chapter 5 includes SD card tests, where we check the capability of the SD card. Chapter 6 includes a cloud portion where we are uploading data to the cloud and doing some operations over the cloud, and the last chapter is about the conclusion of the project.

2 CHAPTER 02 LITERATURE REVIEW

2.1 LITERATURE REVIEW

As the name implies, data acquisition systems (DAQ) are hardware and/or software tools used to gather data in order to monitor or research a phenomenon. As electronic technology develops, data acquisition through electronic devices has improved and become accurate, adaptable, and reliable. Devices for data acquisition should connect to various sensors that identify the phenomenon being studied. The majority of data acquisition systems use various analogue signals produced by transducers to collect data. The majority of applications require some processing of these signals. In order to process analogue signals, an analogue to digital converter (ADC) converts them into a digital format.

Each data acquisition task presents particular challenges. Data acquisition test and measurement techniques can be portable or stationary, used on a test cell or in harsh environments, and in lab research or for educational purposes. These systems could really measure more than just electric signals; they can also measure things like temperature, acceleration, sound, force and pressure, light, position, and displacement. A transducer is needed at the DAQ channels' input. The signal conditioning, DAQ hardware, computer, and software all play a role in getting the best results from a DAQ system.

Battery and energy monitoring is a frequent task needed in the development and research of electric vehicles. But wiring and setting up the data acquisition apparatus can take a lot of time. A wireless sensor system that uses the "Internet of Things" to gather the required data and make it accessible in real-time was developed to make this task more convenient. A wireless sensor network (WSN) with a star topology can be created by joining several sensor units together. Commercially available tools that can measure data and transmit it wirelessly already exist. Examples include wireless multimeters and temperature data loggers that support WLAN.

There are also sensors that measure temperature, humidity, or other analogue signals and transmit data over an Ethernet connection. In the industrial world, monitoring the batteries in electric cars is one potential application for measuring current and voltage with a wired sensor. Wired communication has the drawback of coupling unwanted signals from the environment into the communication wires, which results in noisy measurements. This is particularly crucial for the parametrization and validation processes, which call for precise data free of errors. If there is not enough space or the distances are too great, it may not even be possible to wire a sensor up for communication. Therefore, wired sensor networks require a lot of setup time and may also be very expensive. On the other hand, currently available wireless sensors do not support real-time monitoring or the combination of wireless and wired sensors because they either use sampling rates that are too low for in-depth analyses of dynamic processes. The maximum sampling frequency, channel count, input ranges, ADC resolution, and availability of simultaneous acquisition are the main factors to consider when choosing a data acquisition system. Since the objective is to develop an acquisition system similar to the systems already on the market with a lower production cost, the cost of the device is also crucial. The prices of the devices that are currently on the market vary depending on the factors mentioned as well as some other unique features of each device. The purpose of this work is to develop, implement, and characterize a small data acquisition device that can be powered and controlled by a Flash Drive in order to cut costs. Furthermore, we are designing a system to upload the collected data to the cloud. We will do extensive data analytics over the cloud. In future, we also plan to collect data from many e-bikes simultaneously and provide an uber like ride-sharing services.

3 CHAPTER 03 METHODOLOGY

In this section, we covered complete methodology of our project completions.

3.1 Design

In the design section we design a circuit to interface a set of sensors to the microcontroller. Current sensors are interfaced to measure the current of DC load. It will work on the principle of Hall Effect. To measure current, the magnetic field is measured during the current sensing procedure. Then voltage sensors are interfaced with the Arduino to measure the voltage of the battery source. The voltage in an electrical device can be measured using a variety of circuit types. Resistive type sensors and capacitor type sensors are often employed techniques for sensing voltage. By stringing resistors together in series, resistive type voltage sensors are built. Capacitors are connected in series to create capacitor-type voltage sensors. Then GPS was interfaced with Arduino. Memory module was interfaced with Arduino to store the data from all the sensors.

3.2 Block Diagram

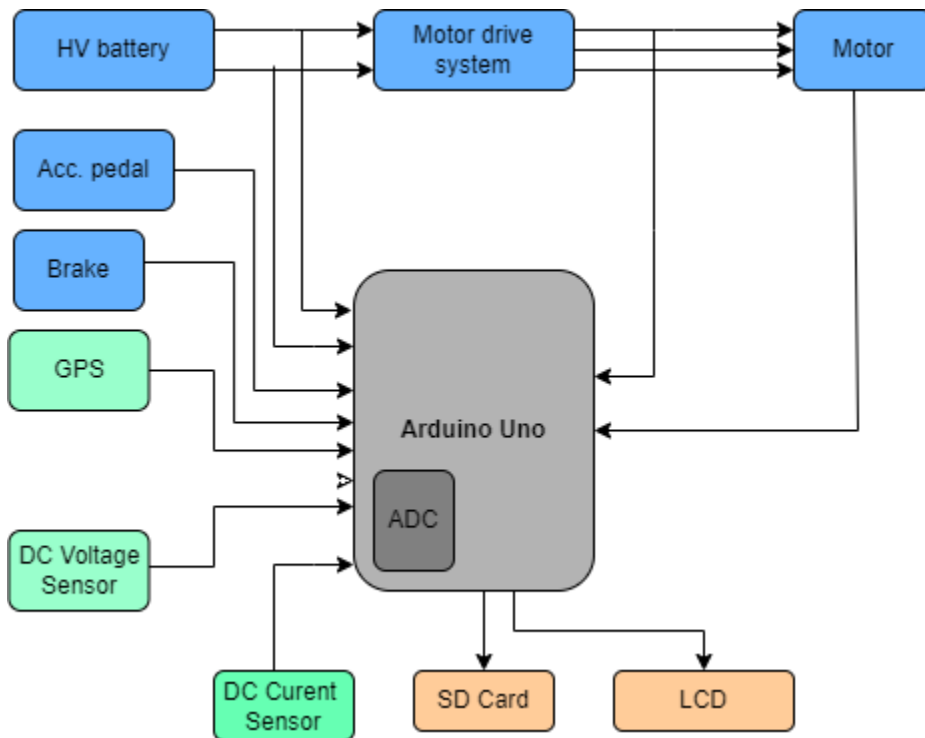


Figure 1 Block diagram

3.3 Flow Chart

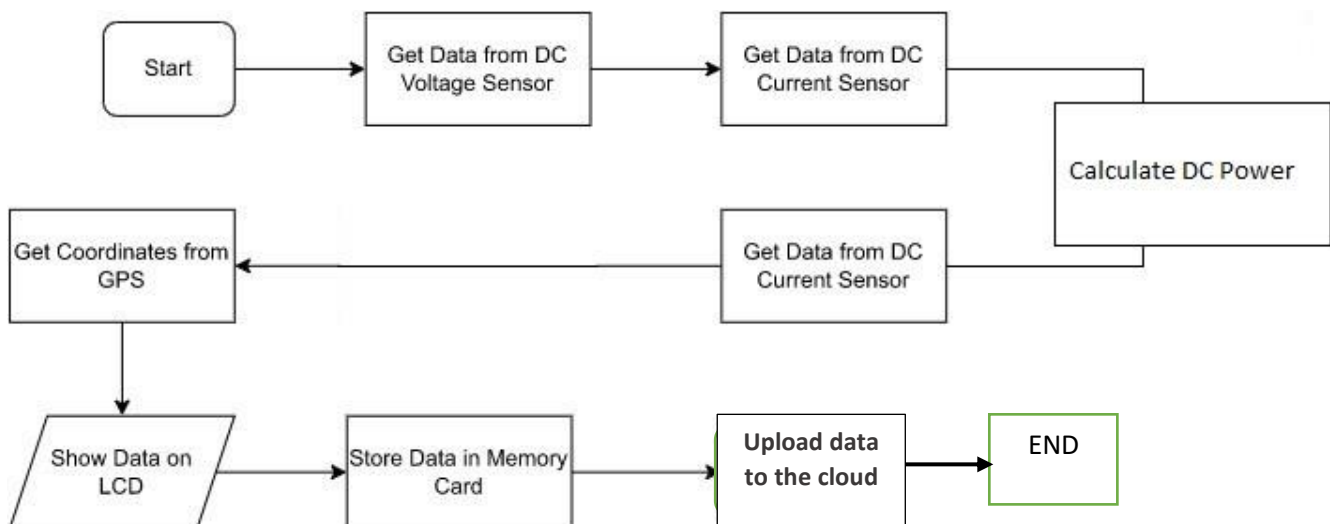


Figure 2 Flowchart

3.5 Implementation

In implementation process the proposed system in this project is implemented in a systematic and organized approach until we reach the final executable system. We divided the implementation process into a number of phases which are followed.

1. In first phase we prepared a proper infrastructure (collection of hardware, software elements) for this project.
2. Then we worked solely on each hardware and software element used in this project.
3. Then we collect and stored the relevant data and results while working on single hardware software element.
4. After getting enough and sufficient knowledge about every single entity used in this project.
5. Finally, we execute the system in an integrated approach and finding a final solution for proposed system.

This project further implemented according to the following flow chart

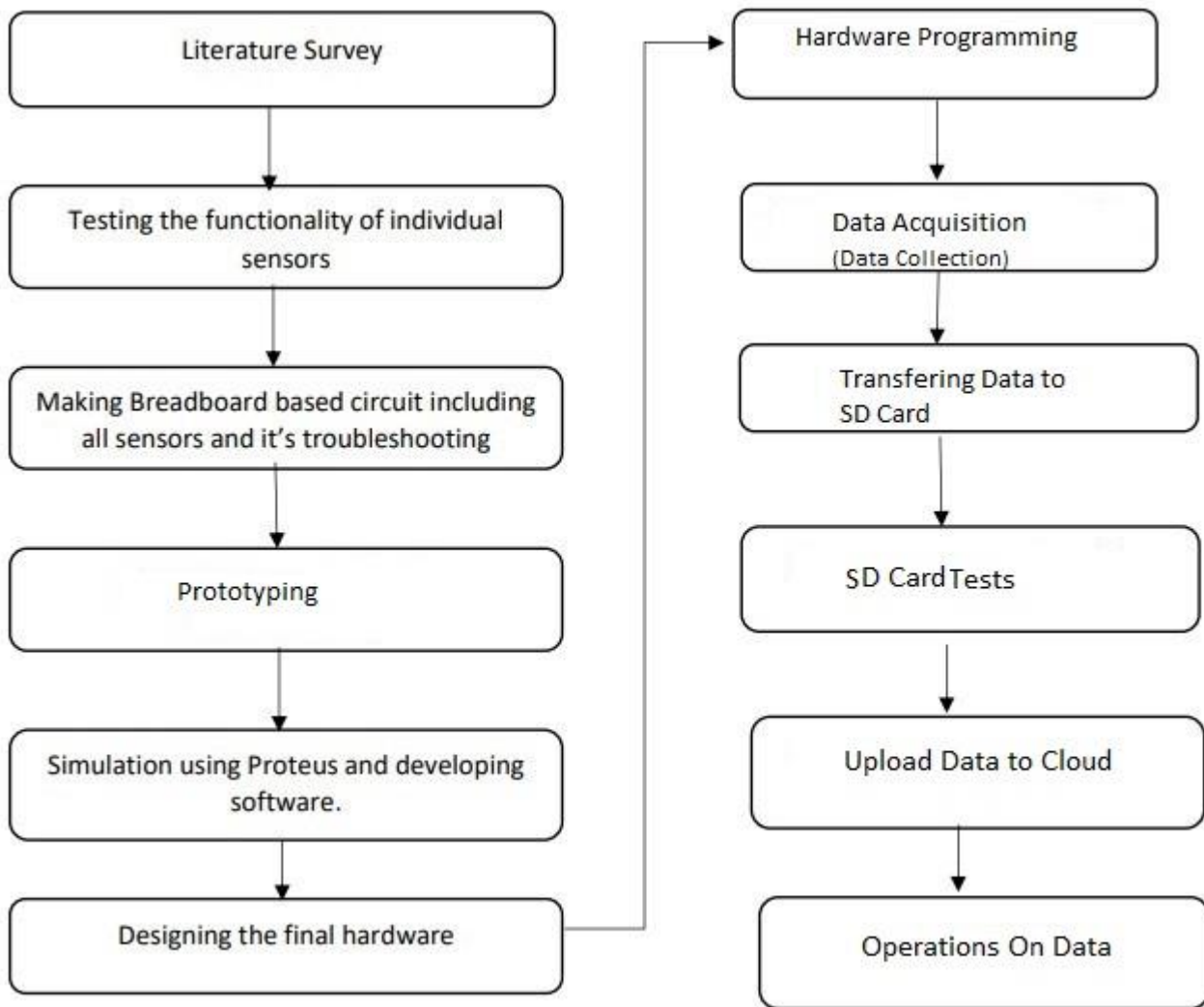


Figure 3 Flowchart of implementation

3.5.1 Testing

Unit testing was done throughout the development process to make sure that all of the modules functioned as expected. Testing of the integrated system will take place once all of the individual components have been constructed, examined, and integrated into the application in question. Both the software and the hardware will be put through their paces by us. And then after that, we will send all of the data to the cloud, along with the data that is sent to an SD card.

3.5.1.1 Prototype

In this phase testing will be performed on the prototype of the data acquisition system.

3.5.1.2 Transferring Data to SD Card

In this part of the tutorial, we will use an interrupt subroutine, build a script consisting of SD card code, and then send data to the SD card.

3.5.1.3 Upload Data to cloud

we use AWS cloud to store data online, but it's too much expensive when we upload data to cloud. but for temporary we use AWS cloud system to manage over data.

3.6 Software & Hardware

The following hardware and software are used in this project

3.6.1 Software

□ Arduino IDE

3.6.1.1 Arduino IDE

Arduino's official software for writing, compiling, and uploading code to Arduino modules/boards. Arduino IDE is open-source software you can download from Arduino's website.

Arduino IDE is open-source software designed by Arduino.cc for writing, compiling, and uploading code to most Arduino modules. Its official Arduino software that makes code compilation so easy that even a non-expert can get begin. It's available for MAC, Windows, and Linux and runs on the Java Platform, which has functions and commands for debugging, editing, and compiling code. There are many Arduino modules, including Arduino Uno, Arduino Mega, Arduino Leonardo, and Arduino Micro. Each board has a microcontroller that accepts coded information. The main code, or sketch, created on the IDE platform generates a Hex File that is uploaded to the board's controller. The IDE environment consists of an Editor for writing code and a Compiler for compiling and uploading code to an Arduino Module. It supports C and C++.

IDE has three main sections.

- Menu Bar
- Text Editor
- Output Pane

When you download and open IDE, you'll see the image below.

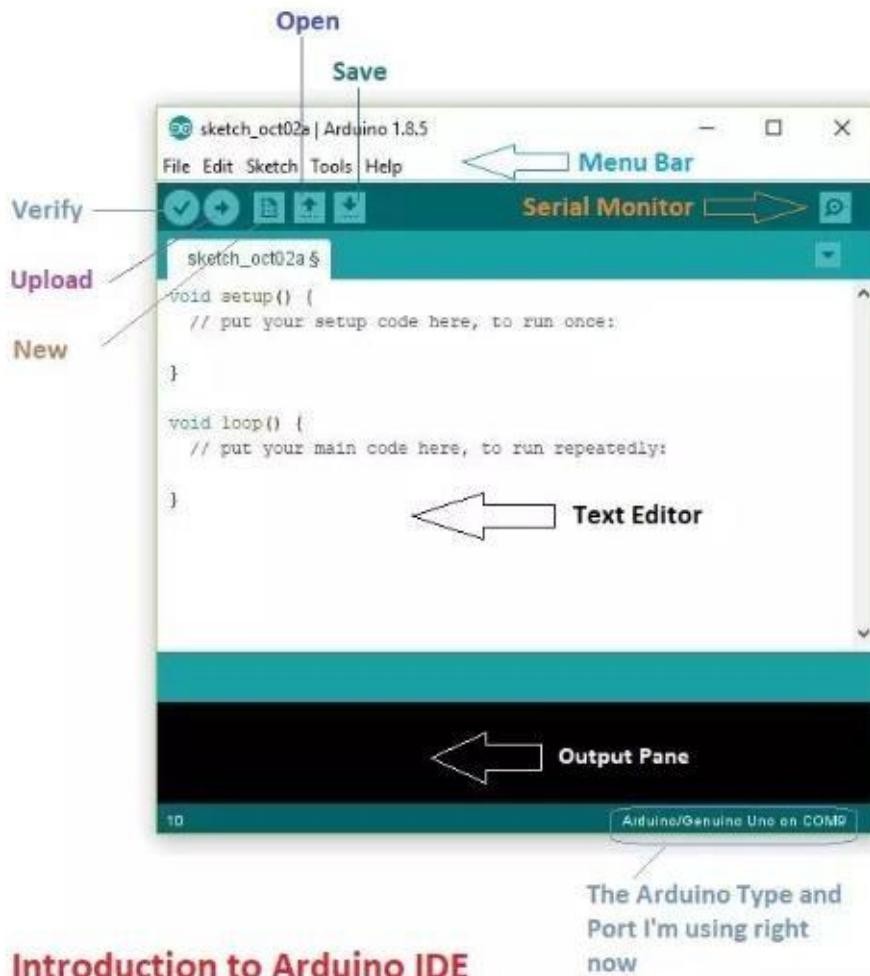


Figure 4 Introduction to Arduino IDE



Figure 5 Menu tab

The button's checkmark verifies the code. After coding, click this.

Arrow key uploads Arduino code.

Dotted paper creates a new file.

The up arrow opens a project.

Downward arrow saves running code.

The top right button is a Serial Monitor, a pop-up window that sends and receives Serial Data. Select Serial Monitor from the Tools panel, or press Ctrl+Shift+M. The Serial Monitor Debugs Sketches so you can see how your programme works. To activate the Serial Monitor, connect your Arduino Module via USB. Select the Arduino Board's baud rate. Assuming your Arduino Uno's Baud Rate is 9600, the output will look like the image below.

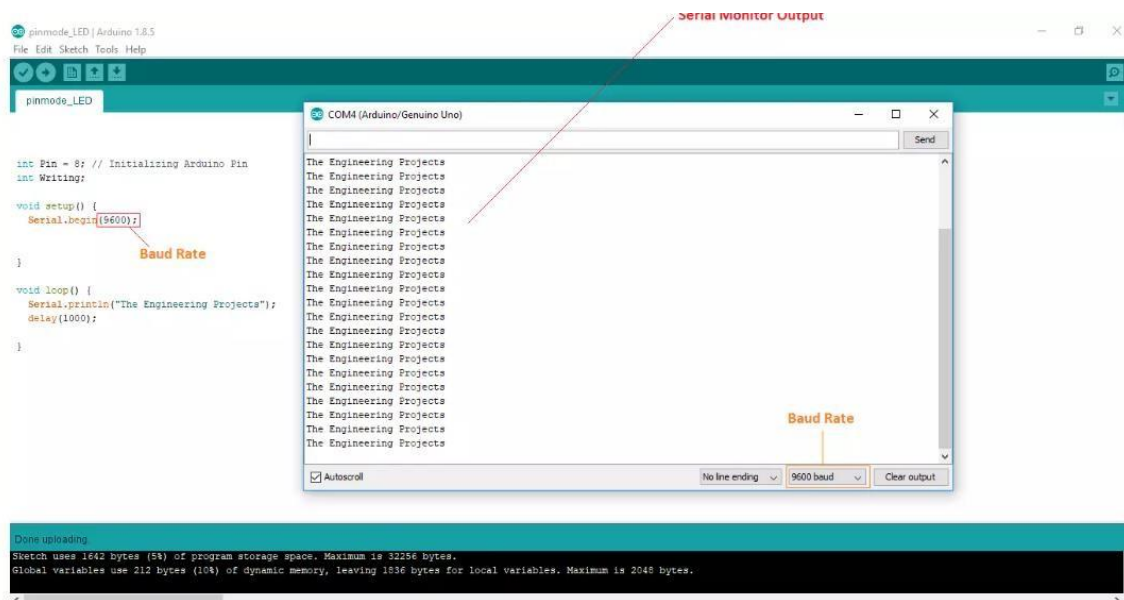


Figure 6 Serial Monitor output

3.7 Hardware

- Arduino Uno Board
- Current Sensing Module (ACS 712)
- DC voltage Sensor
- GPS Module
- LCD Display

- Memory module
- Oscilloscope
- Frequency Generator

3.7.1.1 Arduino Uno Board

Arduino Uno's ATmega328 microprocessor. This board has 14 virtual signal pins, 6 analogue inputs, a 16 MHz ceramic resonator, a USB connection, a power connection, an ICSP beam, and a reset switch. It includes everything we need to start using the microprocessor; simply connect it to a computer via USB or power it with an AC-to-DC inverter or battery. We can play Uno without worrying about mistakes. "Uno" means "one" in European, so it was chosen to celebrate Arduino code. Arduino's Uno board has been replaced by later releases. The Arduino Uno board is the first in a series of USB Microcontrollers and prototypical; the Arduino list of panels has been viewed for a complete list of current, previous, and outdated movements.



Figure 7 Arduino Uno

5 volts 7-12 volts is the atmega328p's functional power (limit). 14 digital input/output pins with PWM (6 of which are output) 6 analogue input/output pins

Arduino Uno programming:

Windows Arduino automates Arduino (IDE). Tools > Arduino > Panel.

Its ATmega328 bootloader is preprogrammed, allowing us to upload Arduino's current code without accessing the hardware interface operator.

Differentiation from other microcontroller:

In contrast to previous panels, the Uno cannot use the FTDI USB-to-serial driver chip. Despite this, it features a USB-to-serial adapter based upon that Atmega16U2 (Atmega8U2 up to version R2).

Communication:

Arduino Uno interfaces with laptops, other Arduino devices, and embedded systems. ATmega328 pins 0 (RX) and 1 (TX) enable the UART TTL (5V) serial call (TX). An onboard ATmega16U2 routes this sequential connection via USB, which appears to PC applications as a digital com port. 16U2 firmware uses standard USB COM drivers, so no extra drivers are needed.

File to get started. The Arduino Software (IDE) has a serial monitor that allows you to communicate and exchange simple textual information between the two Microcontrollers. When data is sent via the USB-to-serial chip and the USB interface to the laptop, the RX and TX LEDs on the board will begin to flash alternately (. Communication protocol on any of the Uno's digital pins is available using the Software Serial library that was included in the Arduino IDE. The ATmega328 also supports I2C (TWI) and SPI interfaces, in addition to the others. The Arduino IDE (IDE) offers the Wire library, which makes it simpler to communicate with the bus.

Data sheet:

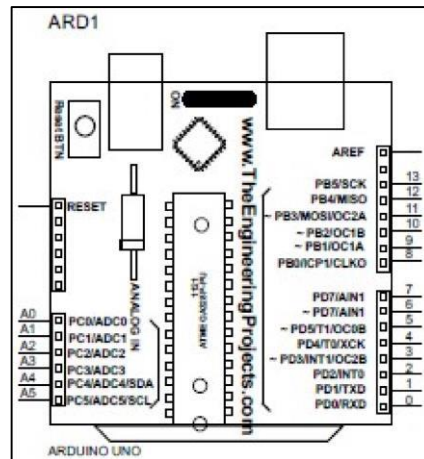


Figure 8 Arduino Uno Circuit Diagram

3.7.1.2 Current Sensing Module (ACS712)

The ACS712 Current Sensor is designed to work with Arduino. Allegro ACS712ELC chips power these sensors. These current sensors measure 5A, 20A, and 30A. We used 5A sensor for our project.

Technical Specifications:

It can detect DC current.

Isolation from the load is provided by this 5A, 20A, and 30A module.

Because it produces analogue voltage, it's simple to combine with MCUs.

When should we utilize the ACS712 module?

In the ACS712 Module, Hall Effect theory and the ACS712 IC monitor current. Because the name of the module derives from the integrated circuit it includes (ACS712), we used the IC directly in our final products rather than the module. ACS712 modules detect AC or DC currents between +5 and -5 amps, +20 and 20 amps, and +30 and -30 amps. Greater range modules reduce precision, so we must choose the right range for our project. This module converts wire current to analogue voltage (0-5V) that can be connected to a

microcontroller. We need to measure current with a microcontroller, so this module works.

The ACS712 module has two green phoenix terminal connections and mounting screws. The wire must travel through these terminals. Since I'm monitoring the load's current, I route the load's cables through the ACS 712 Module. We made sure the module is in series with the load and no shorts occur.

The back has three pins. The module is powered by +5V and the MCU's ground (system). Any microcontroller analogue pin reads the ACS712 module's analogue voltage.

Datasheet:

Characteristic	Symbol	Notes	Rating	Units
Supply Voltage	V_{CC}		8	V
Reverse Supply Voltage	V_{RCC}		-0.1	V
Output Voltage	V_{OUT}		8	V
Reverse Output Voltage	V_{ROUT}		-0.1	V
Output Current Source	$I_{OUT(SOURCE)}$		3	mA
Output Current Sink	$I_{OUT(SINK)}$		10	mA
Overcurrent Transient Tolerance	I_P	100 total pulses, 250 ms duration each, applied at a rate of 1 pulse every 100 seconds.	60	A
Maximum Transient Sensed Current	$I_{PT(max)}$	Junction Temperature, $T_J < T_J(max)$	60	A
Nominal Operating Ambient Temperature	T_A	Range E	-40 to 85	°C
Maximum Junction	$T_J(max)$		165	°C
Storage Temperature	T_{STG}		-65 to 170	°C

Figure 9 ACS 712 Data Sheet

Circuit Diagram:

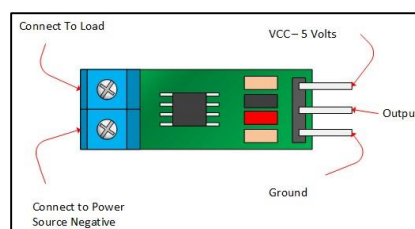


Figure 10 ACS 712 Circuit Diagram

3.7.1.4 DC Voltage Sensor

Current and voltage are needed to measure DC power in this project. We'll measure DC voltage with a resistor-based voltage divider. This sensor measures DC battery power. It measures battery DC voltage.

The bike's 48-volt battery. No 48v DC sensor is available. We made a 50-volt Dc voltage sensor. It can accurately measure 50v.

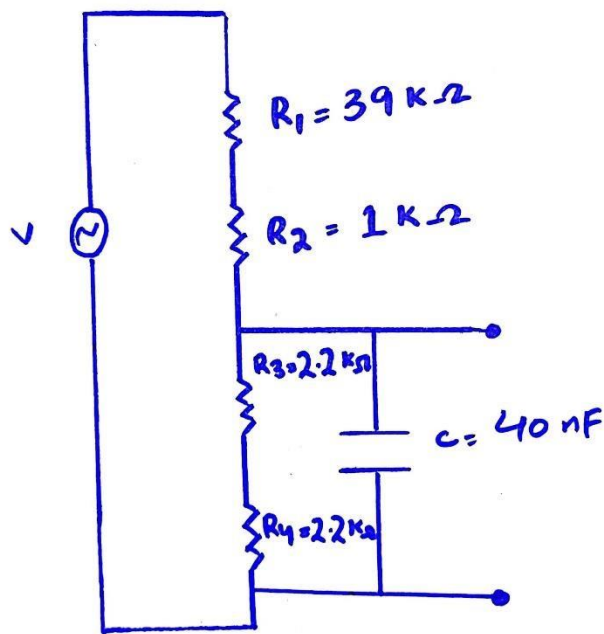


Figure 24 Circuit Diagram of DC voltage sensor

3.7.1.5 GPS Module

GPS navigation devices can receive satellite information to determine a device's location. Using appropriate software, the device can display its location on a map and provide directions. SIM800 has 68 SMT pads and provides all module-to-board hardware interfaces. To detect this, we must: First, a GPS module would store the Electric bike's latitude and longitude.

Features:

- Single supply of 4-7VDC with an asynchronous TTL serial interface operating at 55 mA (typically).
- Data output Baud rate: 9600 bps (Default)
- Format of output that conforms to the NMEA0183 standard
- Standard 4-pin Berg strip interface measuring 2.54mm (0.1") in width. Based on MediaTek's Single-Chip Architecture, the Pitch Module will come equipped with both a normal and right-angled berg strip.
- Patch Antenna Size: 25mm x 25mm x 4mm
- Low Power Consumption: 55mA at acquisition and 40mA at tracking L1 Frequency, C/A coding, 51-channel
- Tracking sensitivity of up to -158 dBm and excellent performance in urban environments
- Accuracy in Position: 3 metres CEP (50 percent) without SA (horizontal)
- Under thirty-six seconds at a Cold Start (Typical)
- The warm up takes less than thirty-four seconds (Typical)
- Under one second is required for the Hot Start (Typical)
- Max. Update Rate: 5Hz (Default: 1 Hz)
- [

Connection

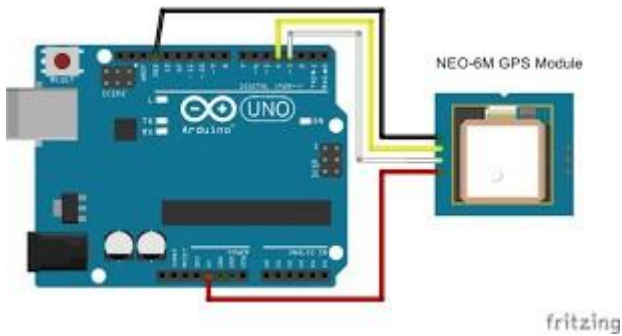


Figure 11 Connection of Arduino to GPS

3.7.1.7 SD Card Module

This project stores data from Current, Voltage, and GPS sensors on an SD card. SD card module used to interface Arduino with SD card.

Micro SD Card Module Pinout

The Micro SD Card module has 6 pins. The Pinout of a Micro SD Card Module is shown below

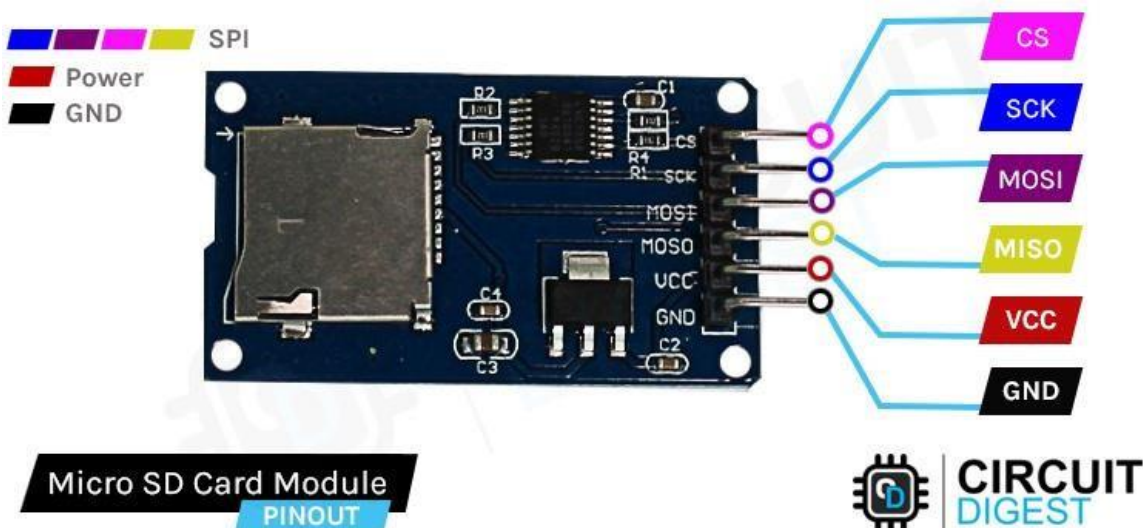


Figure 12 SD Card Module Pins

GND should be connected to the Arduino's ground pin.

VCC is the micro SD card module's 5V or 3.3V power supply pin.

MISO Master in Slave Out. SD Card Module SPI data. Master Out Slave In SD Card Module input pin.

SCK Serial Clock is the Arduino's data synchronization pulse.

CS Chip Select pin allows Arduino to enable or disable module.

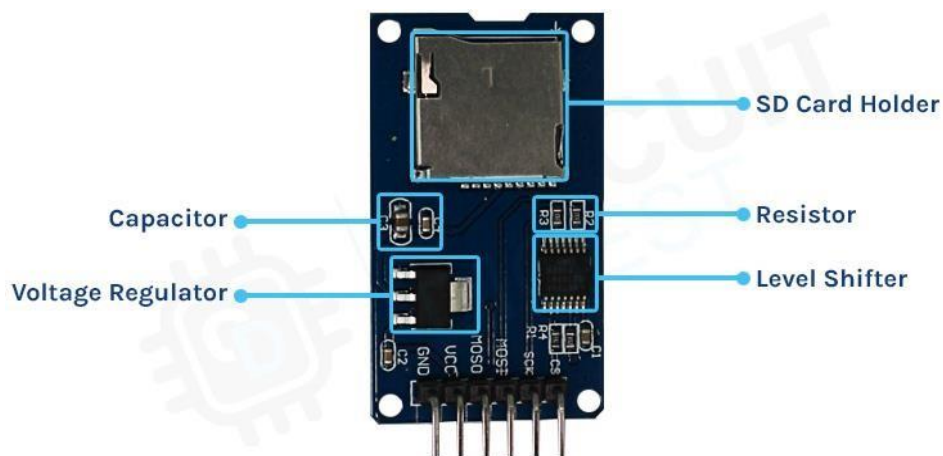


Figure 13 SD Card module parts

Micro SD Card module PCB has few components. First, the Micro SD Card Holder. This holder makes switching SD card modules easy. The second most important thing is the level shifter IC because the SD card module runs only on 3.3V and has a maximum operating voltage of 3.6V, so connecting it to 5V will kill it. The module's ultra-low dropout regulator converts 5V to 3.3V. This module can also run on 3.3V.

Arduino Micro SD Card Module Circuit Connection Diagram

Now that we understand how a Micro SD Card Module works, we can connect all the wires to Arduino and write the code to get sensor data. Micro SD Card Module Arduino Connection Diagram-

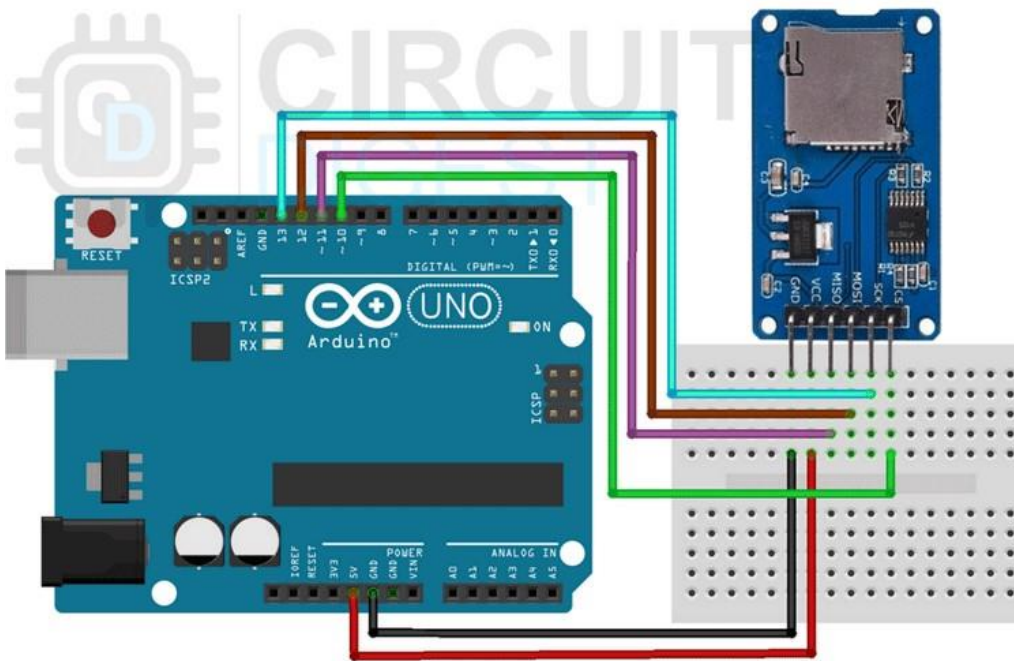


Figure 14 Connection of Arduino to SD card module

Connecting the Micro SD Card module to the Arduino UNO is simple. We just need to connect the SPI lines, SCK, MISO, and MOSI, to the Arduino's SPI lines, SCK(D13), MOSI(D12), and MISO(D11). If there are multiple devices on the SPI bus, we also need to connect the CS line to the Arduino. VCC and Ground power the device.

Preparing the Micro SD Card Module

The SD card reader module can only read FAT16 or FAT32 file systems, so you must format the card before inserting it. If your SD card is new, it's likely factory formatted, which may not work because a pre-formatted card has a FAT file system. Either way, formatting the old card will reduce issues during operation.

Micro SD Module Working

The SD Card module gif is below. After Arduino initialization, the code checks for an SD card. If it finds an SD card, it checks if the text file exists; if not, it creates a new text file and writes testing 1, 2, 3. Text document. Next, the code reads the text file and prints the data to the serial monitor.

Micro SD Module Troubleshooting

- Check module's power supply. Check the onboard LM1117-3.3V regulator.
- Before inserting the Micro SD card, format it.
- When quick format in Windows doesn't work, format the card using regular format or Windows CMD.
- If using an old micro-SD card, clean its contacts..

3.7.1.8 LCD (1602A)

LCD1602 is a dot matrix module that displays letters, numbers, and characters. Each dot matrix position can display one character. Dot pitch and spaces separate characters and lines. 1602 means 2 rows with 16 characters each can be displayed.



Figure 15 LCD

LCD1602 has parallel ports.

Multiple pins simultaneously LCD1602 has 8-port and 4-port connections. If the eight-port connection is used, the Arduino Uno's digital ports are nearly full. No ports are available for more sensors. Therefore, a four-port connection is used.

Pin functions of LCD1602

VSS is grounded; VDD is +5V.

VO: contrast adjustment

RS controls where data is written in the LCD's memory. You can select either the data register, which holds screen data, or an instruction register, where the LCD controller looks for next steps.

R/W: Read/Write pin to select reading/writing mode

E: An enable pin that reads data at High (1). When the signal goes low, the instructions run.

D0-D7: read/write

A and K: Backlight pins. K to GND, A to 3.3v. Open the backlight to see clear text in a dark room.

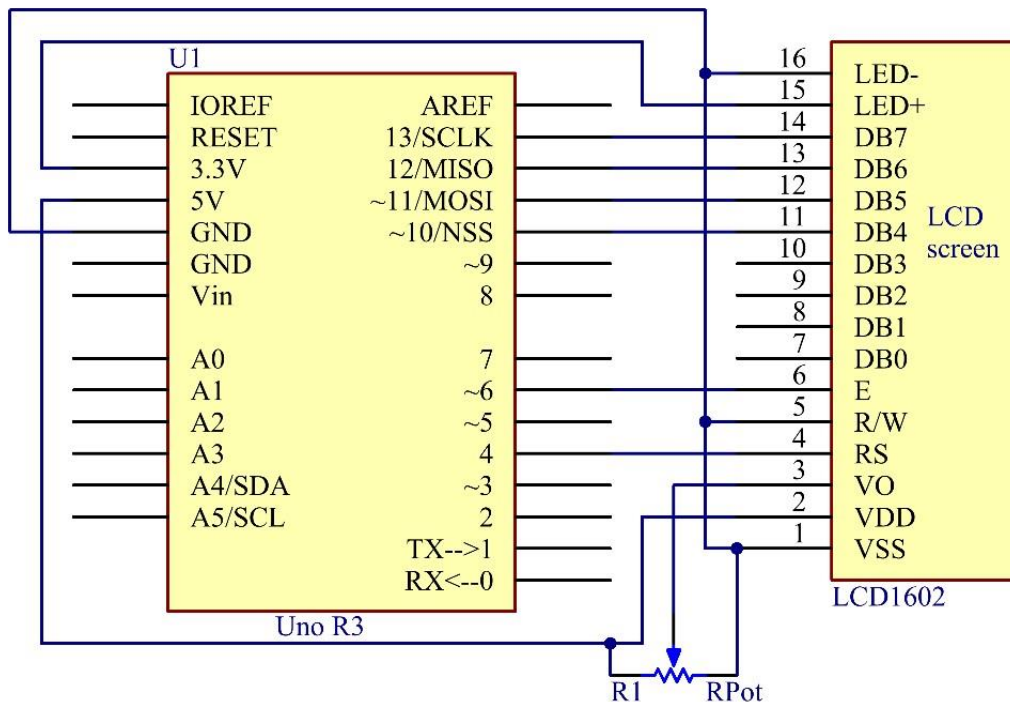


Figure 16 Connection of Arduino to LCD

K to GND and A to 3.3 V turns on LCD1602's backlight. VSS to GND, LCD1602 to power. Connect VO to the potentiometer's middle pin to adjust screen contrast. Connect RS to D4 and R/W to GND to write to LCD1602. Connect E to pin6 and D4-D7 control LCD1602 characters. Function libraries optimize programming.

Hardware oscilloscope

"An oscilloscope, also known as a scope or a two-dimensional plot of one or more signals as a function of time, is a type of electronic test tool that graphically displays changing electrical voltages. The main goals are to display repetitive or single waveforms that would otherwise occur too quickly for the human eye to notice on the screen. Once the waveform has been displayed, various characteristics can be examined, including amplitude, frequency, rise time, time interval, distortion, and more. In the past, calculating these values required manually comparing the waveform to scales integrated into the instrument's screen. These properties can be directly calculated and displayed by modern digital instruments. Science, medicine, engineering, the automotive industry, and the telecommunications industry all use oscilloscopes. For laboratory work and the maintenance of electronic equipment, general-purpose instruments are used.

In this sequence we toggle a signal with the help of Arduino write a pin 8 and then oscilloscope wire are connect to and measure the signal.

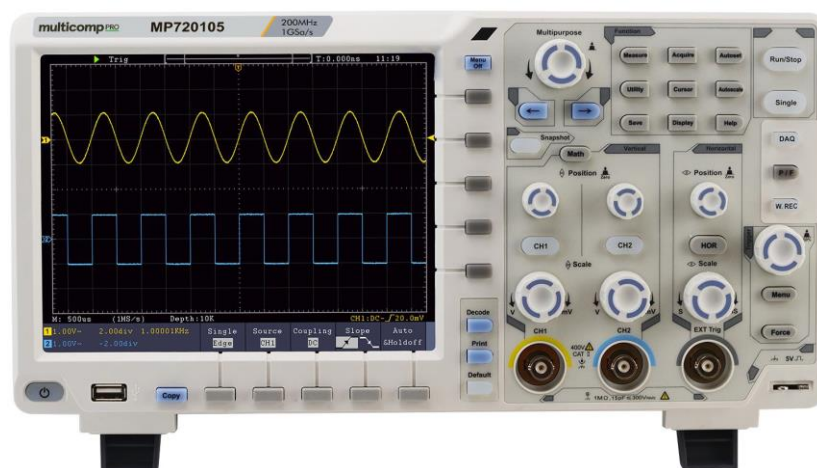


Figure 17 oscilloscope

Signal generator

There is a wide variety of signal generator hardware available on the market today, each of which serves a unique set of needs and fulfils a specific set of functions, and which each ranges in price. These include frequency generators, arbitrary waveform generators, digital pattern generators, function generators, and microwave and RF signal generators. Other types include pitch generators, function generators, and digital pattern generators. In general, there is no device that is appropriate for all of the different applications that could be used.



Figure 18 Signal generator

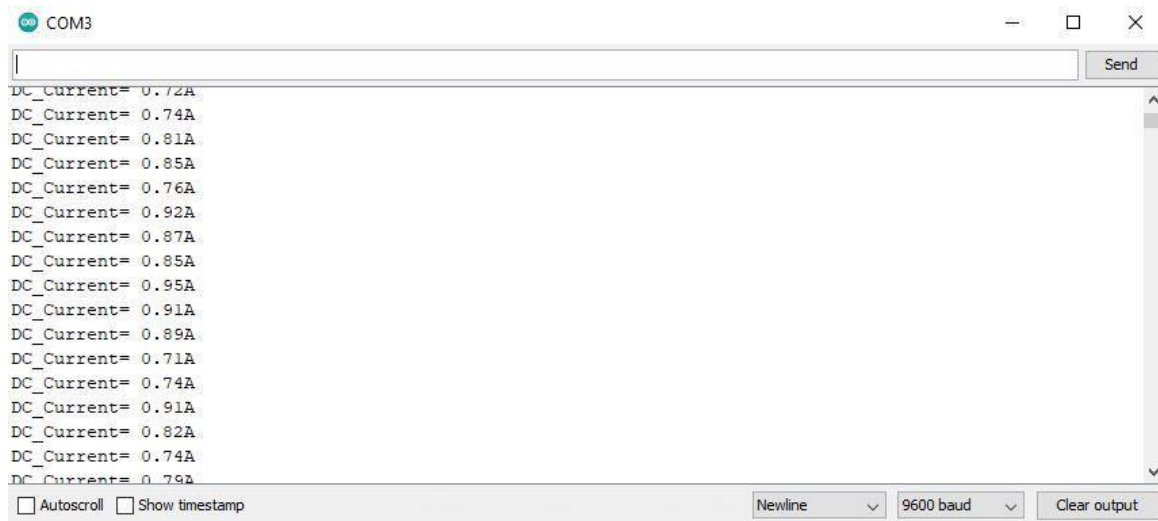
4 Chapter 04 Collecting Data through Sensors

This chapter includes a detailed description of the real-world hardware results, as well as information on its theoretical and practical foundation and the methods utilized to achieve it.

4.1 ACS712 Current Sensor for DC Current

The ACS712 Current Sensor is designed to work with Arduino. Allegro ACS712ELC chips power these sensors. These current sensors measure 5A, 20A, and 30A. We used 5A sensor for our project.

Result on Serial Monitor



Result on LCD

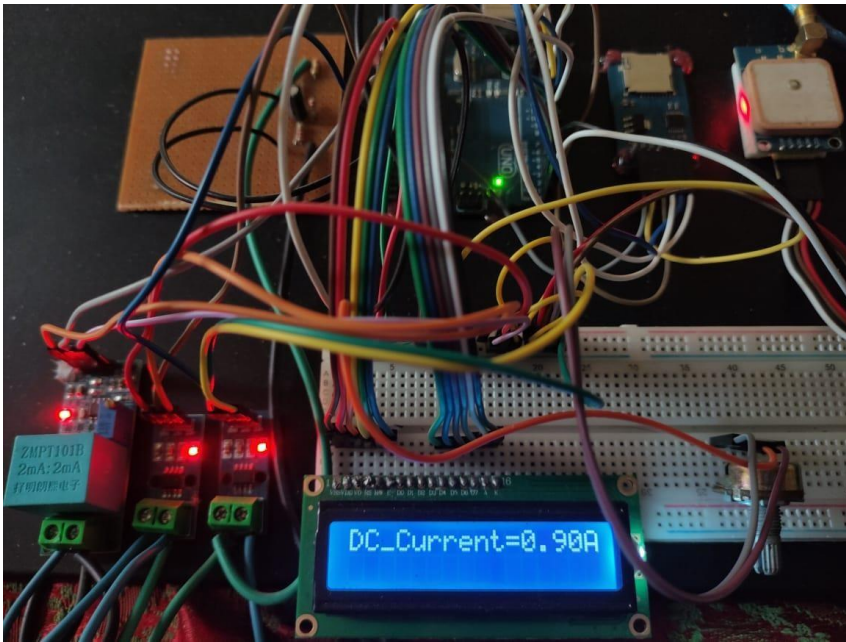


Figure 19 Result of DC Current sensor

4.4 DC Voltage Sensor

The electric bike has a battery that has a voltage of 48 volts. There is currently no DC sensor on the market that can measure a voltage of 48 volts DC. So we used 12 volt Dc voltage sensor.



Result on LCD

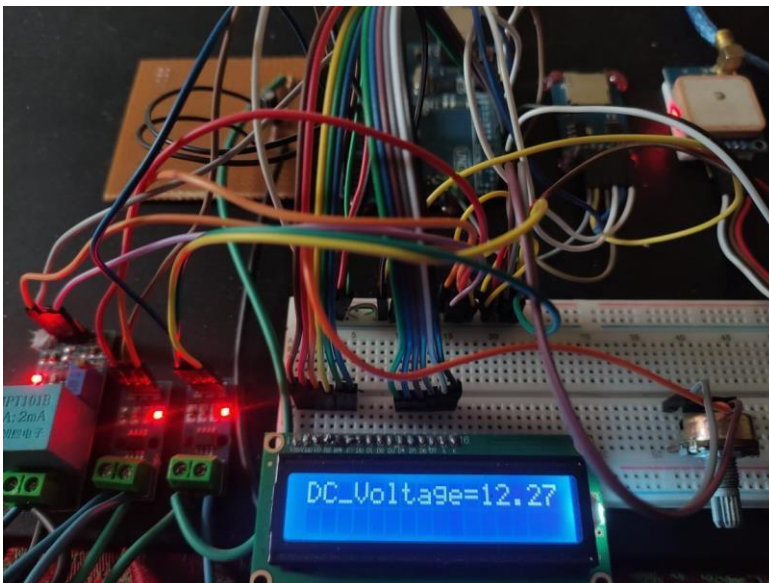
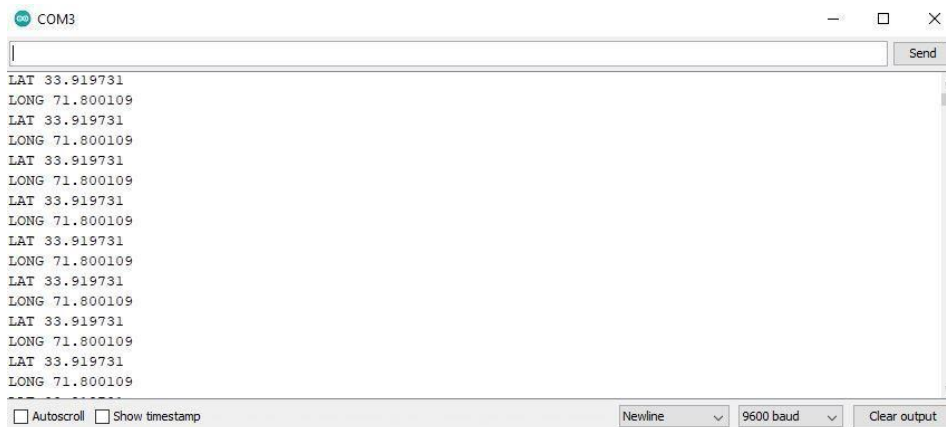


Figure 20 Result of DC voltage Sensor

4.6 GPS

We used a GPS sensor to pinpoint the precise location of the electric bike; initially, we checked it out in software before putting it into action on the hardware. The software and the hardware representations of the location are shown in the following figures.

Result on Serial Monitor



Result on LCD

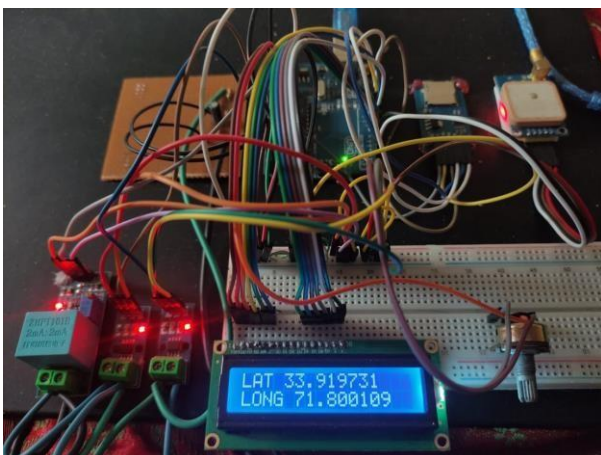


Figure 21 Result of GPS

Result on Map

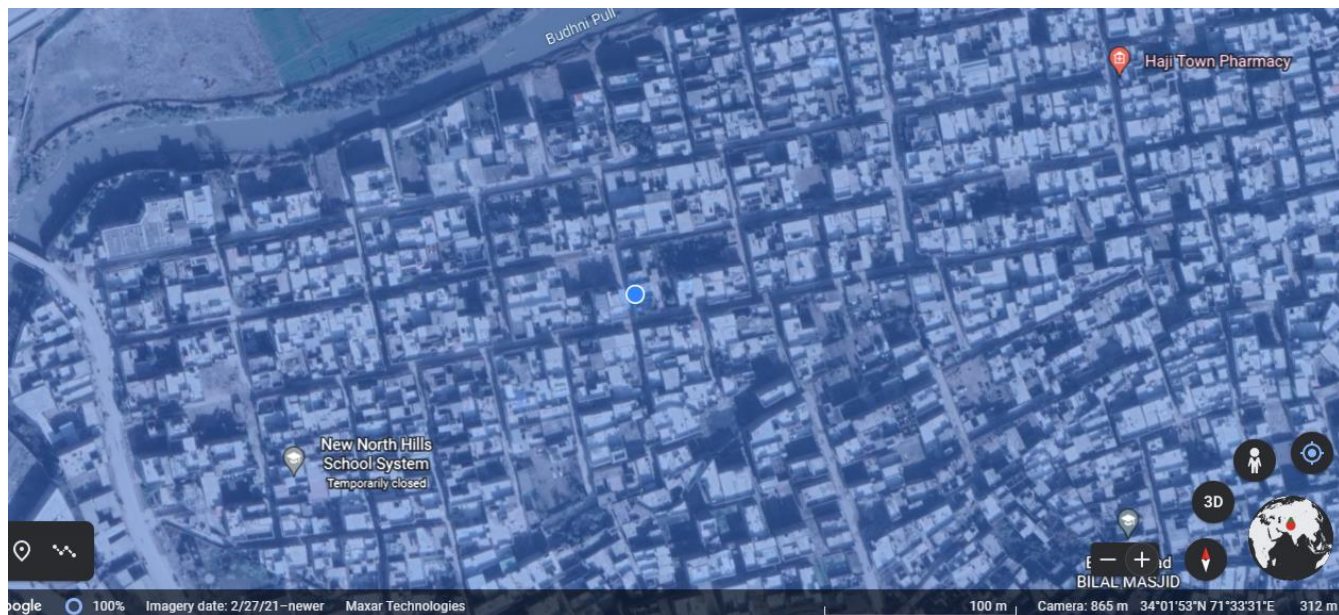


Figure 22 Result of GPS on MAP

4.7 SD Card Module

In the process of working on this project, we will collect data from a variety of sensors, including GPS, Current, and Voltage sensors, and store it on an SD card. We will be using an SD card module as the interface between Arduino and the SD card. The following is a summary of the consequences that resulted from storing data.

Result in Serial Monitor

```
COM3
|
Beginning initialization of SD card:SD initialized Successfully
-----
File already exists . it has been over written
-----
writing to E-bike.txt:
DC_voltage= 12.27v
DC_Current= 0.90A
Ac_voltage= 228v
Ac_Current= 0.47A
DC_Power= 9w
Ac_Power= 107w
LAT 33.919731
LONG 71.800109
DC_voltage= 11.78v
DC_Current= 0.84A
Ac_voltage= 220v
Ac_Current= 0.48A
DC_Power= 10w
Ac_Power= 105w
LAT 33.919731
LONG 71.800109
DC_voltage= 12.27v
DC_Current= 0.90A
Ac_voltage= 228v
Ac_Current= 0.47A
DC_Power= 9w
Ac_Power= 107w
LAT 33.919731
LONG 71.800109
```

Data stored in SD Card

The screenshot shows a Notepad window titled "TESTING - Notepad". The menu bar includes "File", "Edit", "Format", "View", and "Help". The text area contains 255 lines of blue text, each representing an IP address: "100.00200.00300.400500600700". The final line is highlighted with an orange background. The status bar at the bottom indicates the cursor is at "Ln 255, Col 28", the zoom is "100%", the encoding is "Windows (CRLF)", and the language is "UTF-8".

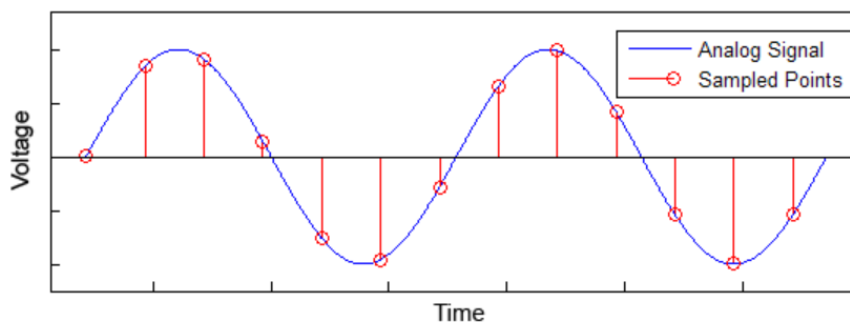
Line	IP Address
1	100.00200.00300.400500600700
2	100.00200.00300.400500600700
3	100.00200.00300.400500600700
4	100.00200.00300.400500600700
5	100.00200.00300.400500600700
6	100.00200.00300.400500600700
7	100.00200.00300.400500600700
8	100.00200.00300.400500600700
9	100.00200.00300.400500600700
10	100.00200.00300.400500600700
11	100.00200.00300.400500600700
12	100.00200.00300.400500600700
13	100.00200.00300.400500600700
14	100.00200.00300.400500600700
15	100.00200.00300.400500600700
16	100.00200.00300.400500600700
17	100.00200.00300.400500600700
18	100.00200.00300.400500600700
19	100.00200.00300.400500600700
20	100.00200.00300.400500600700
21	100.00200.00300.400500600700
22	100.00200.00300.400500600700
23	100.00200.00300.400500600700
24	100.00200.00300.400500600700
25	100.00200.00300.400500600700
26	100.00200.00300.400500600700
27	100.00200.00300.400500600700
28	100.00200.00300.400500600700
29	100.00200.00300.400500600700
30	100.00200.00300.400500600700
31	100.00200.00300.400500600700
32	100.00200.00300.400500600700
33	100.00200.00300.400500600700
34	100.00200.00300.400500600700
35	100.00200.00300.400500600700
36	100.00200.00300.400500600700
37	100.00200.00300.400500600700
38	100.00200.00300.400500600700
39	100.00200.00300.400500600700
40	100.00200.00300.400500600700
41	100.00200.00300.400500600700
42	100.00200.00300.400500600700
43	100.00200.00300.400500600700
44	100.00200.00300.400500600700
45	100.00200.00300.400500600700
46	100.00200.00300.400500600700
47	100.00200.00300.400500600700
48	100.00200.00300.400500600700
49	100.00200.00300.400500600700
50	100.00200.00300.400500600700
51	100.00200.00300.400500600700
52	100.00200.00300.400500600700
53	100.00200.00300.400500600700
54	100.00200.00300.400500600700
55	100.00200.00300.400500600700
56	100.00200.00300.400500600700
57	100.00200.00300.400500600700
58	100.00200.00300.400500600700
59	100.00200.00300.400500600700
60	100.00200.00300.400500600700
61	100.00200.00300.400500600700
62	100.00200.00300.400500600700
63	100.00200.00300.400500600700
64	100.00200.00300.400500600700
65	100.00200.00300.400500600700
66	100.00200.00300.400500600700
67	100.00200.00300.400500600700
68	100.00200.00300.400500600700
69	100.00200.00300.400500600700
70	100.00200.00300.400500600700
71	100.00200.00300.400500600700
72	100.00200.00300.400500600700
73	100.00200.00300.400500600700
74	100.00200.00300.400500600700
75	100.00200.00300.400500600700
76	100.00200.00300.400500600700
77	100.00200.00300.400500600700
78	100.00200.00300.400500600700
79	100.00200.00300.400500600700
80	100.00200.00300.400500600700
81	100.00200.00300.400500600700
82	100.00200.00300.400500600700
83	100.00200.00300.400500600700
84	100.00200.00300.400500600700
85	100.00200.00300.40050060

Figure 23 Data Stored in SD Card

5 Analog to Digital Conversion & SD Card Testing

5.1 Sampling

The signals generated by the electric bike are analogue signals. To process these signals in computers, we need to convert the signals to "digital" form. While an analogue signal is continuous in both time and amplitude, a digital signal is discrete in both time and amplitude. A process called sampling is used to convert a signal from continuous time to discrete time. The value of the signal is measured at certain intervals in time. Each measurement is referred to as a sample. The reasons for converting analogue signals to digital signals are that we are using this data for processing and doing extensive analysis on it. We cannot process and store analogue signals. Therefore, we will have to convert these signals into a form that can be processed and stored.



. This shows the way that a given analog signal might be sampled. The frequency at which the signal is sampled is known as the sampling rate.

Figure 24 Sampling rate

5.1.1 Timers and Interrupts

Timer interrupts **allow us to perform a task at a specific time interval regardless of what else is going on in your code**. In this chapter, we will explain how we setup and execute an interrupt in Clear Timer on Compare Match or CTC Mode. Normally when we write an Arduino sketch the Arduino performs all the commands encapsulated in the loop () {} function in the order that they are written, however, it's difficult to time events in the loop (). Some commands take longer than others to execute, some depend on conditional statements (if, while...) and some Arduino library functions (like digital Write or analogRead) are made up of many commands. Arduino timer interrupts allow us to momentarily pause the normal sequence of events taking place in the loop () function at precisely timed intervals, while we execute a separate set of commands. Once these commands are done the Arduino picks up again where it was in the loop ().

Table 16-5. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clk _{IO} /1 (No prescaling)
0	1	0	clk _{IO} /8 (From prescaler)
0	1	1	clk _{IO} /64 (From prescaler)
1	0	0	clk _{IO} /256 (From prescaler)
1	0	1	clk _{IO} /1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

Figure 25 Prescaler and the Compare Match Register

We will control the speed of the timer counter incrementation by using something called a prescaler. A prescaler dictates the speed of our timer according the following equation:

$$(\text{timer speed (Hz)}) = (\text{Arduino clock speed (16MHz)}) / \text{prescaler}$$

So a 1 prescaler will increment the counter at 16MHz, to select 1 prescaler we need to set CS10 , an 8 prescaler will increment it at 2MHz, a 64 prescaler = 250kHz, and

so on. As indicated in the tables above, the prescaler can equal 1, 8, 64, 256, and 1024.

Table 16-4. Waveform Generation Mode Bit Description⁽¹⁾

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1X at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Figure 26 Registers table

We use the registers mentioned in above tables and we design an Arduino program which go to the Interrupts service routine when the value of Compare Match Register and Output Compare Register become equal and take a sample, after that the program clear the value using CTC Mode (Compare Time Counter).

5.1.2 Code Structure

```
Untitled script
1
2 #include <SPI.h>
3 #include <SD.h>
4 void setup()
5
6 {
7   noInterrupts();          // disable all interrupts
8
9   TCCR1A = 0;
10
11   TCCR1B = 0;
12
13   TCNT1 = 0;
14
15   OCR1A = 1;              // compare match register 16MHz/256/2Hz
16
17   TCCR1B |= (1 << WGM12); // CTC mode
18
19   TCCR1B |= (1 << CS11);  // clk/8 prescaler
20
21   TIMSK1 |= (1 << OCIE1A); // enable timer compare interrupt
22   interrupts();           // enable all interrupts
23
24 }
25
26 ISR(TIMER1_COMPA_vect)    // timer compare interrupt service routine
27
28 {
29   // ISR body
30 }
31 void loop()
32
33 {
34   // your program here...
35 }
```

5.1.3 Timer ISR ()

We determine the time. After how much time it takes to come into the interrupt service routine, we defined an output pin and every time the programme comes into the ISR, the pin toggles. After that, we find the average time of the trigger level using an oscilloscope. The purpose of doing this is that, we are finding the ADC ISR () time and Timer ISR () to find which one is faster and how much. Below is the ISR () function which is doing nothing, just toggle a pine to find the time.

ISR (TIMER1_COMPA_vect)

```
{
    DigitalWrite (8, !digitalRead(8));
}
```

Result

We calculate that the programme takes an average of 12.5us to reach the interrupt service routine and toggle a pin.

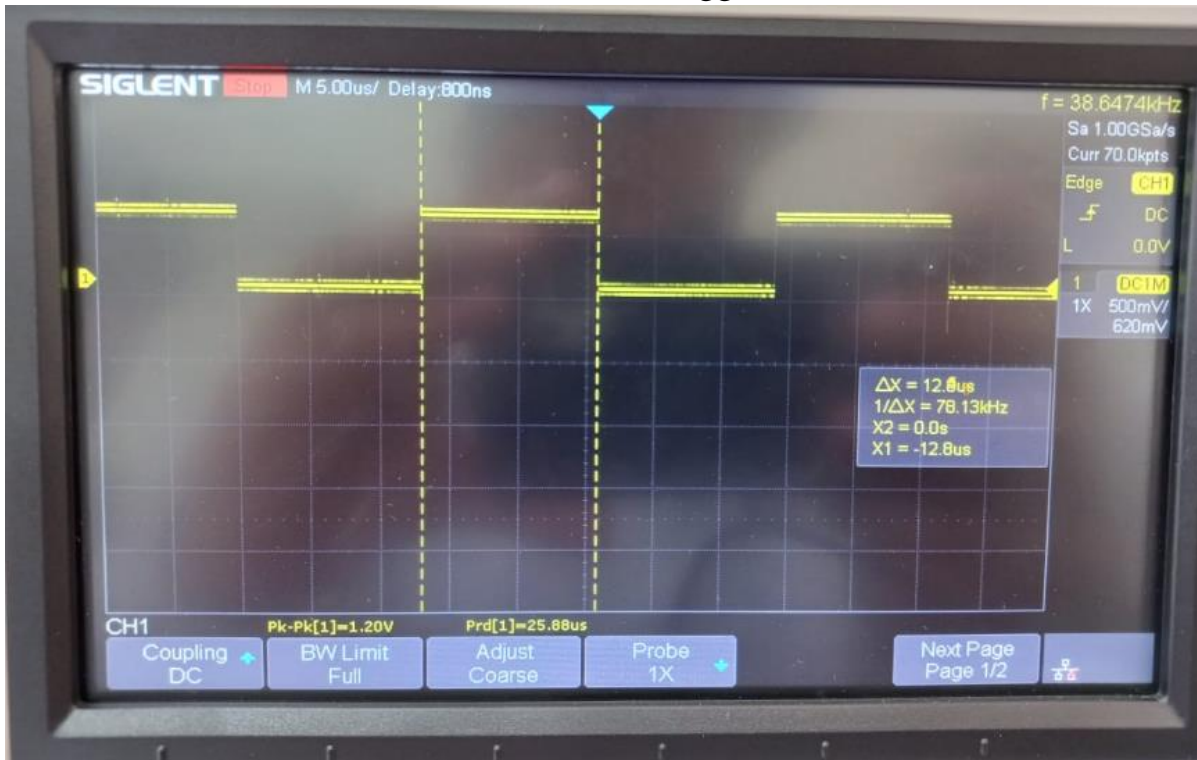


Figure 27 Timer ISR

5.2 Digitizing Analog Signals

An analogue signal is a correct representation of a signal. But, we cannot store the complete waveform of an analogue signal because it is a continuous signal and it has infinite values. It is therefore very costly in terms of storage and time consumption. Due to this, we use the sampling concept to store limited values from the analogue signals.

Any digital processor, such as a computer, needs digital values for processing, transporting, and storing data. When a signal is converted into a number, there is a lot that can be done with it. Calculations, analysis, manipulations, translations, encoding, encrypting, etc. are possible only when the data is in digital form.

An analogue to digital converter (ADC) is used to convert any analogue signal into digital data, which makes it easier to process and store, as well as more precise and reliable by minimising errors. The Arduino Uno has six on-board ADC channels which can be used

to read analogue signals in the range of 0 to 5V. It has a 10-bit ADC, which means it will give digital values in the range of 0–1023 (2^{10}).

5.2.1 Analog to Digital Conversion

analogRead () -

The microcontroller Arduino UNO has a circuit inside called an analog-to-digital converter or ADC that reads this changing voltage in the range of 0 to 5V and converts it to a digital number between 0 and 1023 as Arduino UNO has a 10-bit ADC. When the shaft is turned in one direction, 0 volts are going to the pin, and the input value is 0. When the shaft is turned in the opposite direction, 5 volts are going to the pin and the input value is 1023.

With Arduino Uno, we can use six pins for digital input, A0 to A5. By connecting electronic components here for the input, we can read how much voltage is applied. Shown in the figure below:

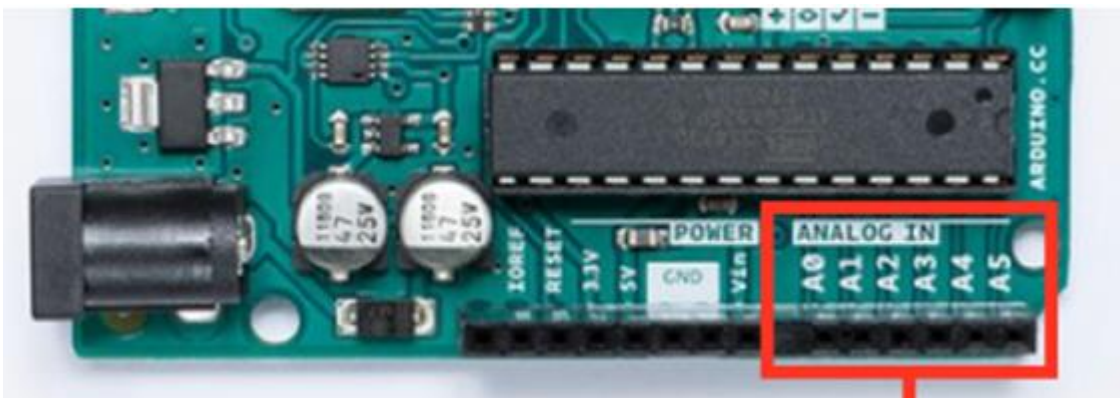


Figure 28 Analog pins

5.2.2 ADC ISR ()

We are reading data through analog pins (A0 to A5) and converting this data to digital within the ISR () function. We set the OCR value to the lowest, which is '1', TCNT to '0', and select the 1 Prescaler by setting the CS10 register. We are determining the time at which the values are converted to digital in ISR (). Furthermore, we will compare the time required for data conversion to digital in the ISR (ADC time) and the average time taken by the program to go to the ISR ().

Arduino program

```
Untitled script
1 #include <SPI.h>
2
3 int countC = 1;
4 void setup()
5 {
6   pinMode(8, OUTPUT);
7   pinMode(A0, INPUT);
8   pinMode(A1, INPUT);
9   pinMode(A2, INPUT);
10
11   noInterrupts();           // disable all interrupts
12   TCCR1A = 0;
13   TCCR1B = 0;
14   TCNT1 = 0;
15   OCR1A = 1;               // compare match register 16MHz/256/2Hz
16
17   TCCR1B |= (1 << WGM12);   // CTC mode
18   TCCR1B |= (1 << CS10);    // 1 prescaler
19   TIMSK1 |= (1 << OCIE1A);  // enable timer compare interrupt
20   interrupts();             // enable all interrupts
21 }
22
23 ISR(TIMER1_COMPA_vect) {    // timer compare interrupt service routine
24   digitalWrite(8, !digitalRead(8));
25   if (countC <= 10000){
26     countC++;
27     float Ia = analogRead(A0);
28     float Ib = analogRead(A1);
29     float Ic = analogRead(A2);
30   }
31 }
32 void loop()
33 {
34   // your program here...
35 }
36 }
```


5.2.3 Results

We calculate that it takes the program an average of 13.0 μs to go to the interrupt service routine, convert an analogue signal to digital data (ADC), and toggle a pin.

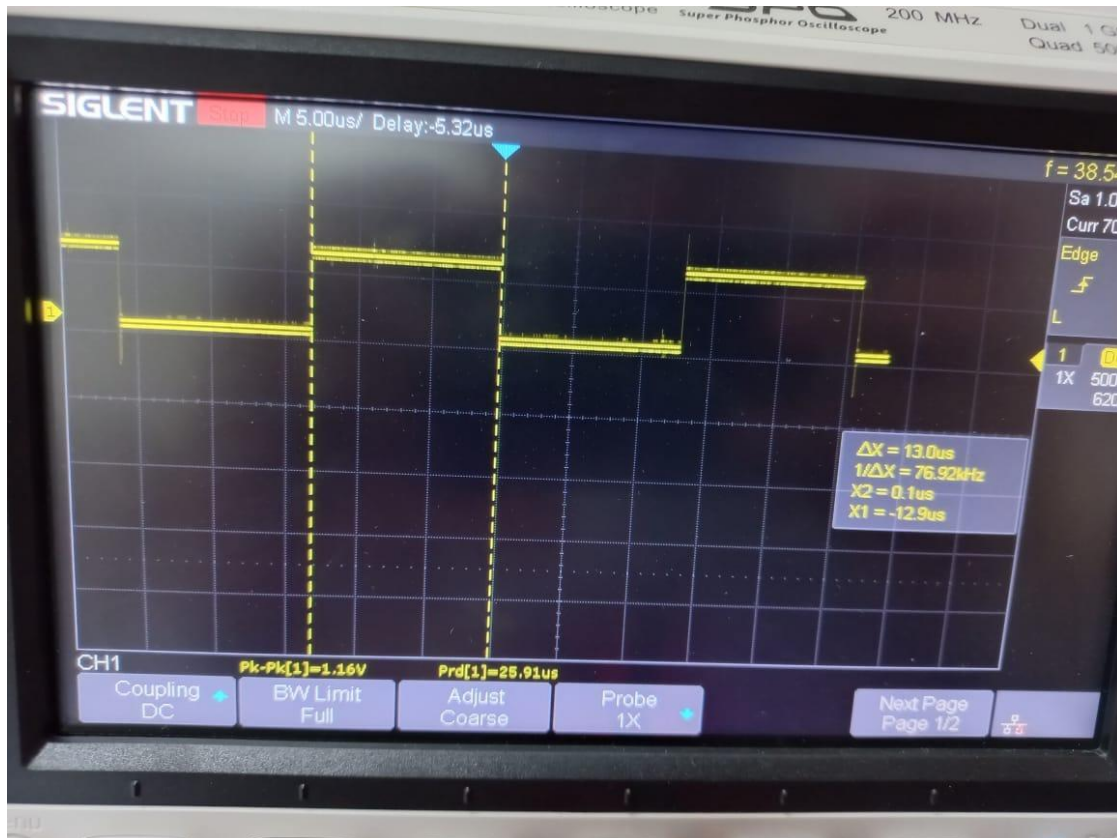


Figure 29 ADC time

5.3 SD Card Testing

In this project, we are storing 10,000 samples per second on an SD card. This is a huge amount of data. Therefore, we are doing some tests on the SD card to check whether the SD card is capable of storing this much data in one second or not. Furthermore, we will do some extra tests to minimise the time and storage costs and optimise the space well.

We are storing sensor data which is in analogue form. Consequently, we are converting sensor data to digital data using an ADC.

5.3.1 Examine for Missing Cycles

We are transferring a huge amount of data to the SD card. Therefore, the first test we have done is to check for the missing cycles in the data. To determine this, we sent a data set of 10,000 records to the SD card. After that, we checked the data on the SD card and, luckily, there were no missing cycles. We received the exact data on the SD card that we had sent. We are sending this data to the cloud for extensive analysis, so it is important to make sure that the data is correct and well formed.

Result

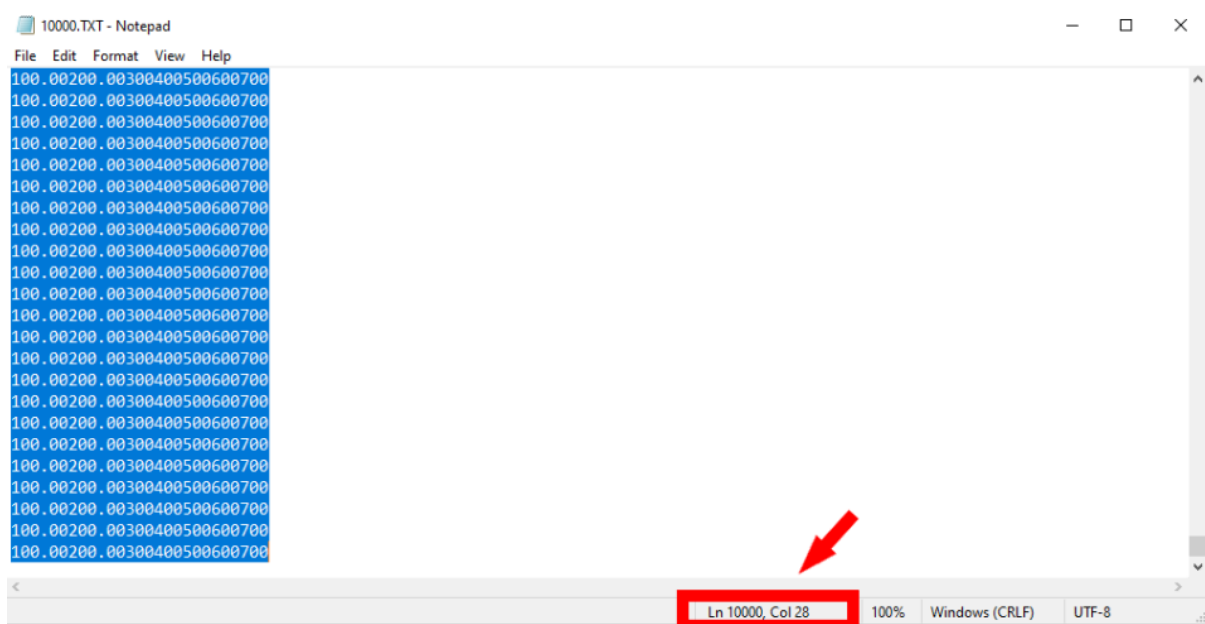


Figure 30 Result(Missing cycles)

5.3.2 Separate Data Transfer Speed

Separate data transfer means that we transfer each sensor's data in an individual print () function to the SD card. The average time taken is **13.4us** for transferring 18 bytes of data to an SD card. We are using five print functions as we are transferring five sensors' data. In the text test, we will concatenate the data into just one print function. Of these two methods, we will go with the one that takes a small amount of time to transfer data to the SD card.

Oscilloscope Result

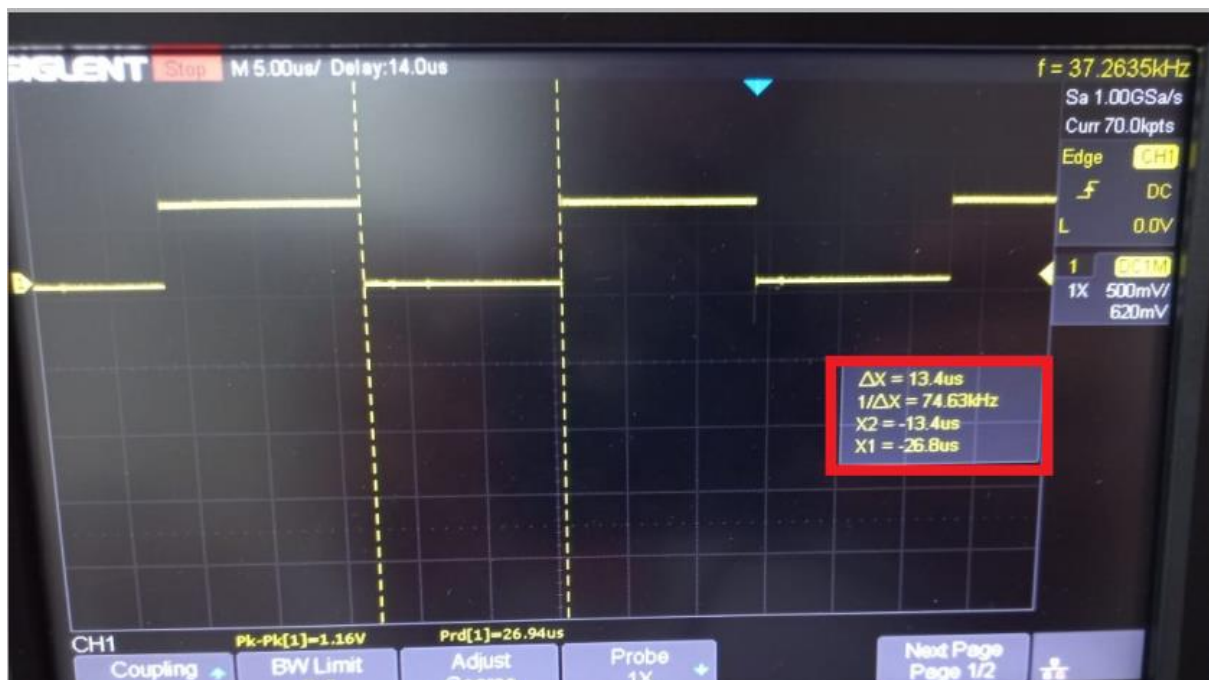


Figure 31 Transfer speed

5.3.3 Concatenated Data Transfer Speed

In this test, we concatenate all the sensor's data and transfer it using one print () function only. After transferring, we determine that the average time required to transfer data in one print function is 17.2us. Therefore, we will not use this method because its time is much greater than the previous method.

Oscilloscope Result



Figure 32 transfer speed

5.3.4 Time Taken: Analog to Digital Conversion & Data Transfer to SD Card

As we know, Analog to Digital Conversion takes an average time of 13.0us to convert analogue data to digital in ISR. Furthermore, we transfer the converted digital data to an SD card and determine that the total time taken for ADC and transferring to the SD card is 17.2 u.

Oscilloscope Result



Figure 33 transfer speed

5.4 Conclusions

Following this investigation and analysis, we conclude that the SD card is easily capable of storing 10,000 samples per second. Therefore, we can proceed with the model we designed for this project. We also determine that there are no missing cycles. We received the exact data on the SD card that we had sent.

After determining the results of 5.3.2 and 5.3.3, we came to know that we have to send each sensor's data in a separate print function instead of concatenating them.

The most important point of this chapter is that the time taken by the programme to come to the ISR is greater than the time taken by ADC to convert the data to digital.

ISR timer () = 12.5us

ADC timer in ISR = 13.04us

Therefore, timer ADC = Timer ADC in ISR - Timer ISR ()

6 UPLOADING DATA TO CLOUD & SOME OPERATIONS

6.1 DESIGN

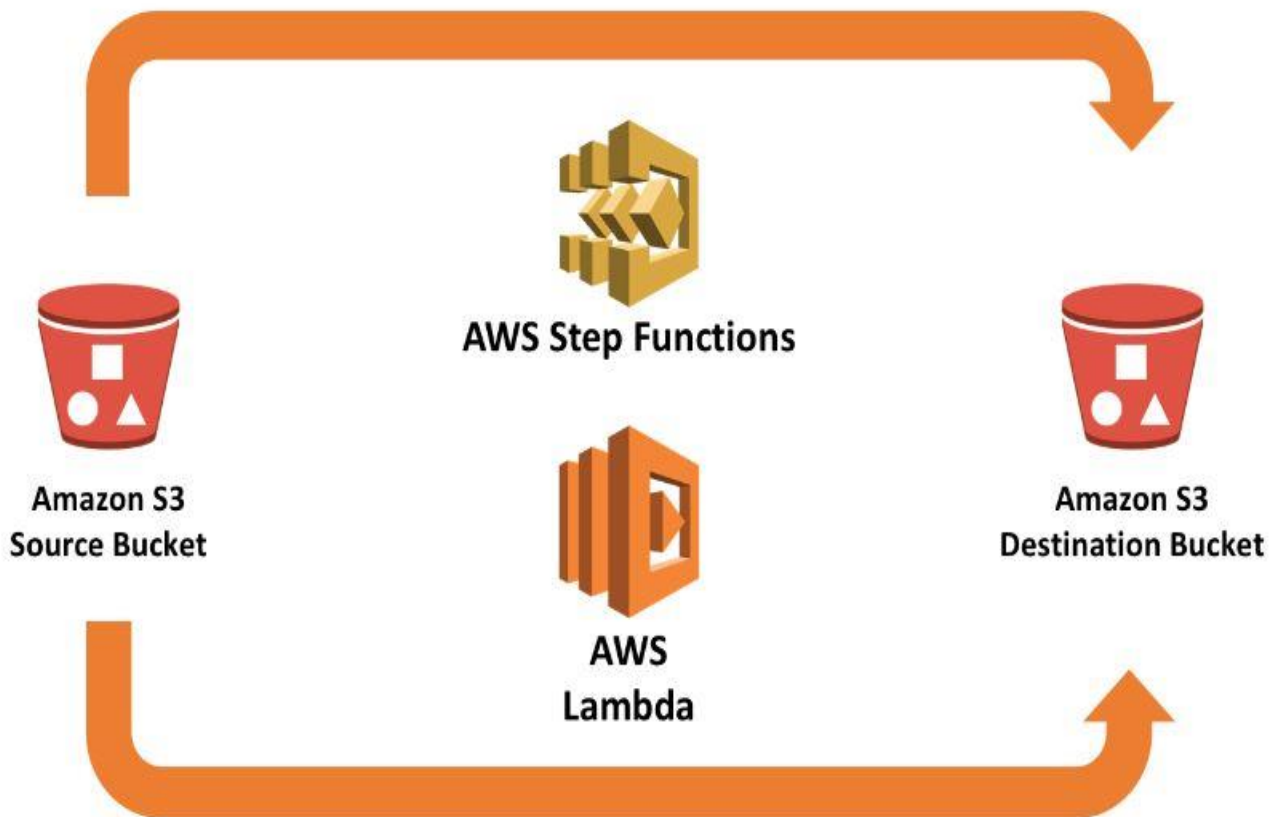


Figure 34 Cloud Work Design

6.2 S3 BUCKETS CREATION

6.2.1 Introduction

S3 stands for simple storage service, and it is AWS's cloud storage service. S3 provides the ability to store, retrieve, access, and back up any amount of data at any time and place.

We will create two s3 buckets, a source bucket and a destination bucket. A source bucket will receive data from any source, i.e. mobile application, local hard drive, etc. While the destination bucket will store the data after the lambda function does some operations on it.

6.2.2 Source Bucket

Let's begin by creating an empty S3 bucket. All we have to do is go to the S3 page from your AWS console and click on the "Create bucket" button. Make sure to leave the "Block all public access" checkbox ticked and click on "Create bucket"

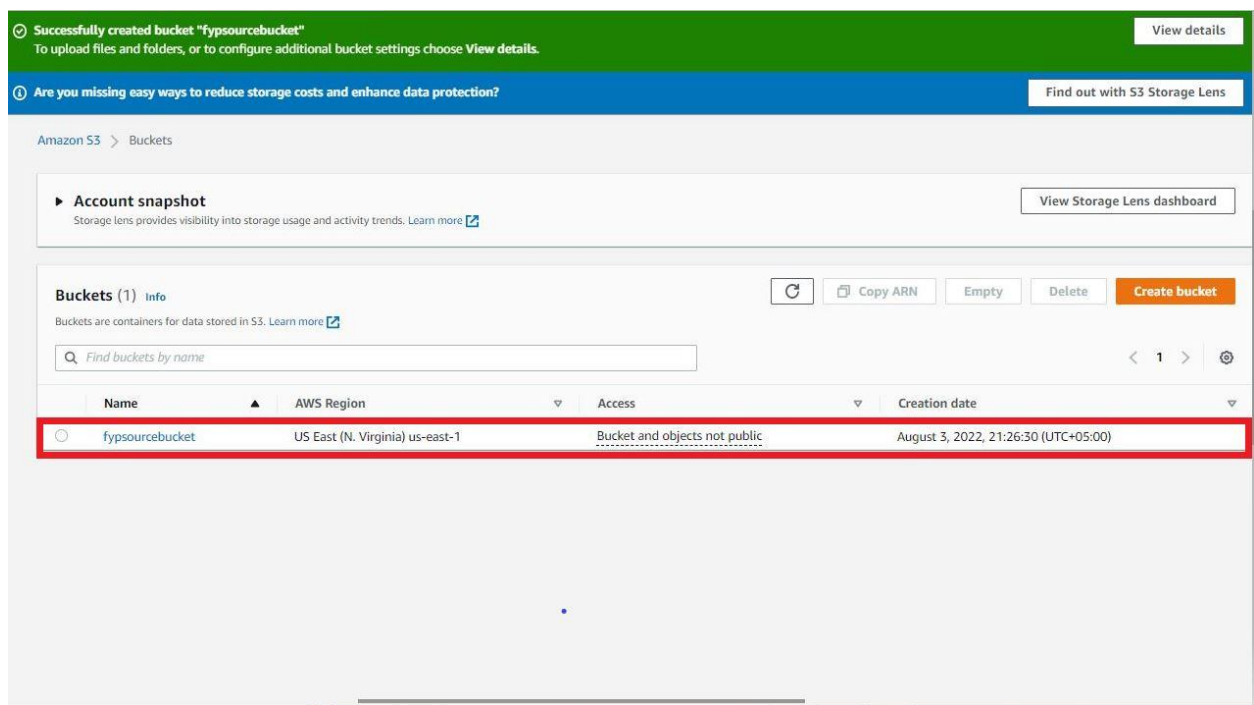


Figure 35 Source bucket

6.2.3 Destination Bucket

Now, building an empty S3 destination bucket. All we have to do is to go to the S3 page from the AWS console and click on the "Create bucket" button. Leave the "Block all public access" checkbox unchecked for the destination bucket because only administration will have the access to access and use the files stored in the destination bucket and click on "Create bucket"

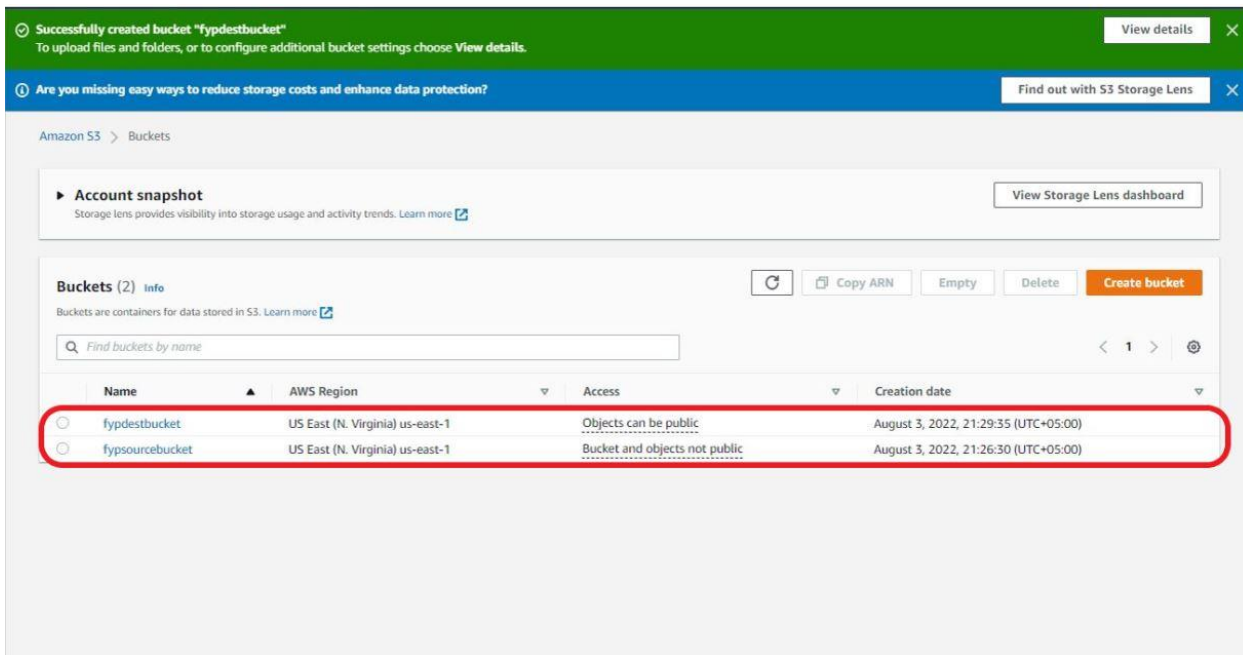


Figure 36 Destination bucket

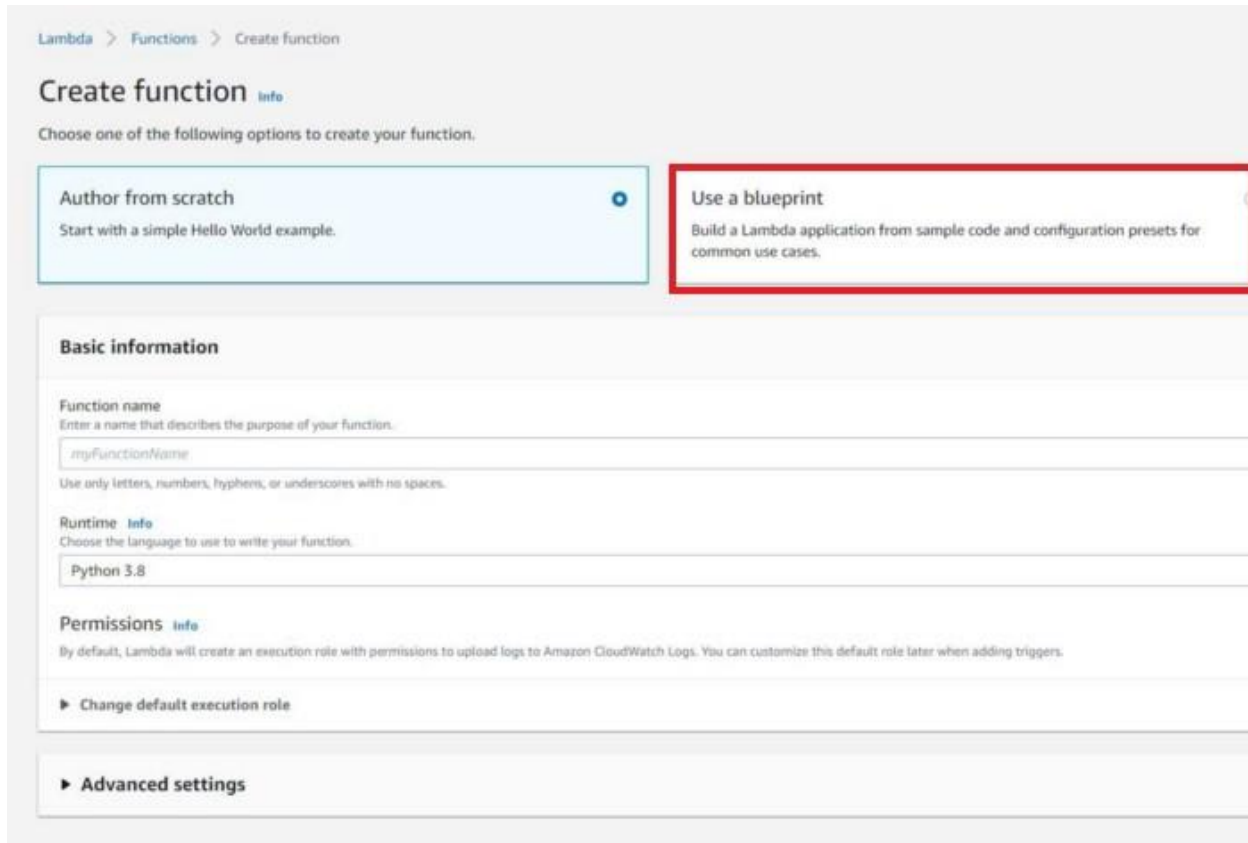
6.3 LAMBDA FUNCTION

6.3.1 Introduction

Lambda is a compute service that allows us to run code without provisioning or managing servers. Lambda runs our code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, and logging. With Lambda, we can run code for virtually any type of application or backend service. All we need to do is supply our code in one of the languages that Lambda supports.

6.3.2 Creating lambda function

From the Services tab on the AWS console, click on “Lambda”. From the left pane on the Lambda page, select “Functions” and then “Create Functions”.



The screenshot shows the AWS Lambda 'Create function' page. At the top, there's a breadcrumb trail: 'Lambda > Functions > Create function'. The main heading is 'Create function' with an 'Info' link. Below this, it says 'Choose one of the following options to create your function.' There are two options: 'Author from scratch' (with a blue circle icon) and 'Use a blueprint' (with a red box around it). The 'Use a blueprint' option has a description: 'Build a Lambda application from sample code and configuration presets for common use cases.' Below the options, there's a 'Basic information' section with a 'Function name' field (placeholder: 'myFunctionName') and a 'Runtime' dropdown (selected: 'Python 3.8'). There's also a 'Permissions' section with a link to 'Change default execution role' and an 'Advanced settings' section.

We select "Use a blueprint" and give the function a suitable name, "fyplambdafunction". Since I'll be using Python 3.8, I chose "Python 3.8" as the runtime language. There are other versions of Python2 and Python3 available as well. We select a runtime language and click on the "Create function" button. From the list of Lambda functions on the "Functions" page, we select the function we just created and we will be taken to the function's page.

Lambda automatically creates an IAM role for us to use with the Lambda function. The IAM role can be found under the "Permissions" tab on the function's page. **We need to ensure that the function's IAM role has permission to access and/or manage the AWS services we connect to from inside our function. Make sure we add "S3" permissions to the IAM role's list of permissions, accessible via the IAM console.**

6.3.3 Attach policy to IAM role

To attach policies to the IAM role named "operationOnData" we have created. We need to search for "IAM" in AWS's console and go to the "Roles" under "IAM". After that, in roles, we need to select our role named "operationOnData" and add permissions to the role.

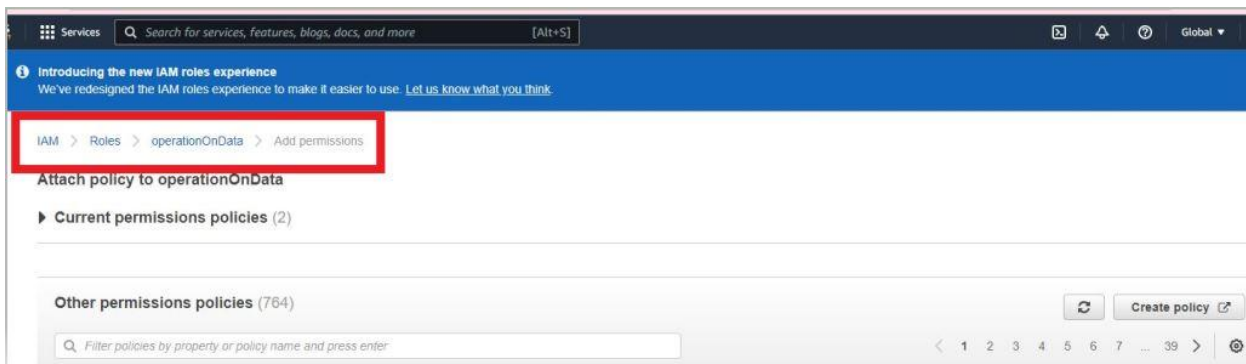


Figure 37 IAM permissions

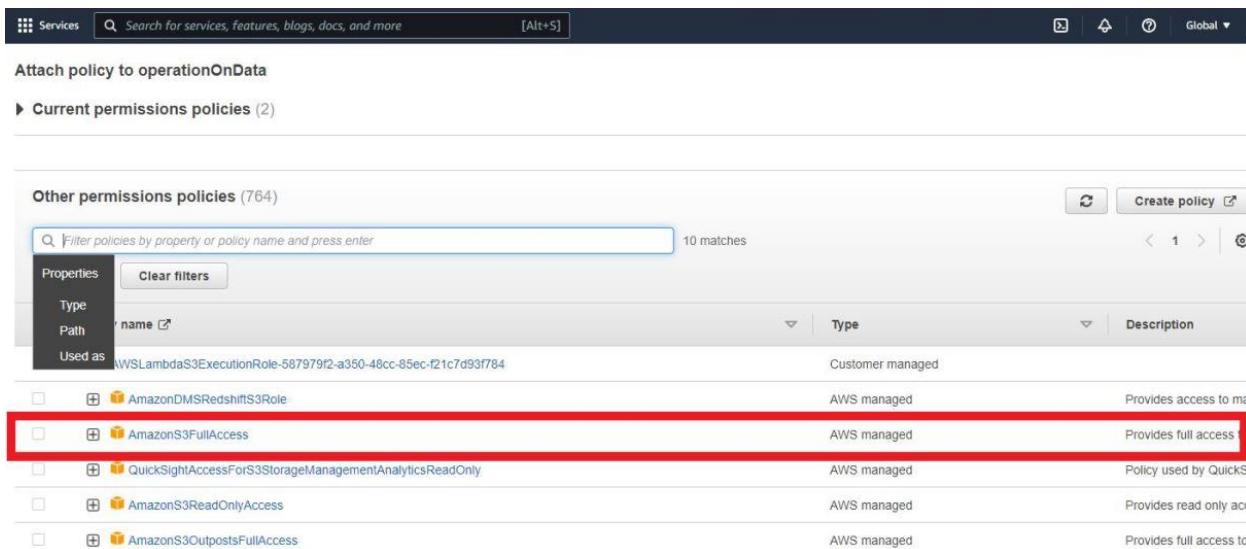


Figure 38 S3FullAccess

6.3.4 Adding a trigger for our Lambda function

We want the Lambda function to be invoked every time a file is uploaded to the “Source Bucket”. To do this, we will use an S3 bucket PUT event as a trigger for our function. Under the “Designer” section on our Lambda function’s page, click on the “Add trigger” button.

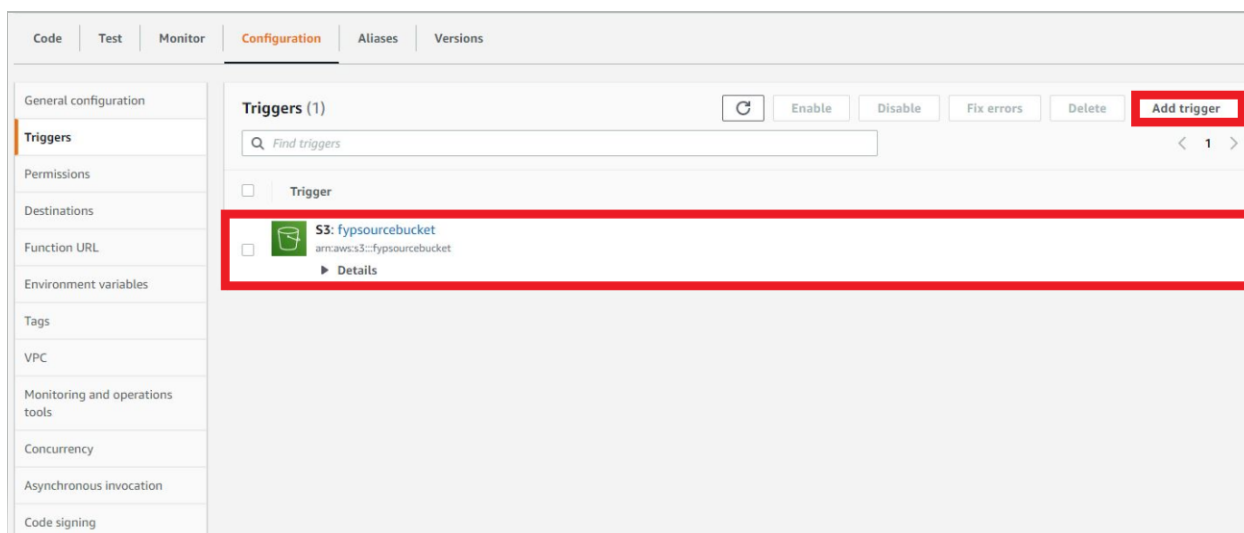


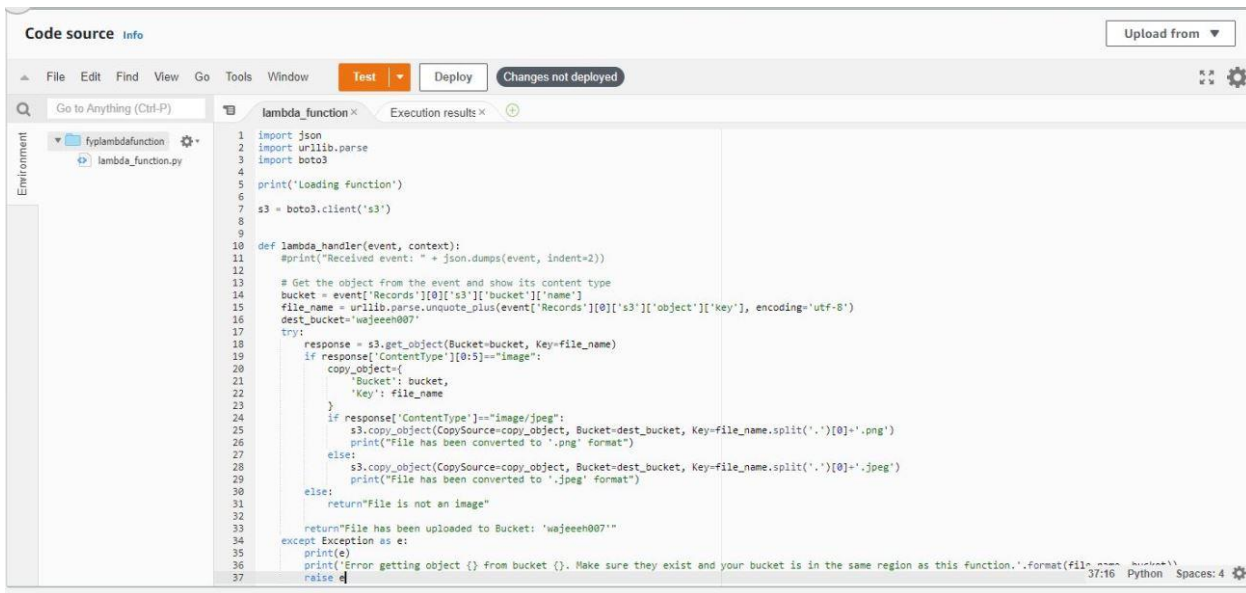
Figure 39 S3 trigger

6.3.5 Adding code to our Lambda function

There are 3 ways we can add code to your Lambda function:

1. Through the code editor available on the console.
2. By uploading a .zip file containing all your code and dependencies.
3. By uploading code from an S3 bucket.

We will use the first method for our project. On function page, go down to the “Function code” section to find the code editor.

The image shows a code editor interface with a menu bar (File, Edit, Find, View, Go, Tools, Window) and buttons for 'Test' and 'Deploy'. A status bar at the top right says 'Changes not deployed'. On the left, an 'Environment' pane shows a project named 'fypsourcebucket' with a file 'lambda_function.py'. The main editor area displays the following Python code:

```
1 import json
2 import urllib.parse
3 import boto3
4
5 print('Loading function')
6
7 s3 = boto3.client('s3')
8
9
10 def lambda_handler(event, context):
11     #print("Received event: " + json.dumps(event, indent=2))
12
13     # Get the object from the event and show its content type
14     bucket = event['Records'][0]['s3']['bucket']['name']
15     file_name = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'], encoding='utf-8')
16     dest_bucket = 'wajeeh007'
17
18     try:
19         response = s3.get_object(Bucket=bucket, Key=file_name)
20         if response['ContentType'][:5] == "image":
21             copy_object = {
22                 'Bucket': bucket,
23                 'Key': file_name
24             }
25             if response['ContentType'] == "image/jpeg":
26                 s3.copy_object(CopySource=copy_object, Bucket=dest_bucket, Key=file_name.split('.')[0] + '.png')
27                 print("File has been converted to '.png' format")
28             else:
29                 s3.copy_object(CopySource=copy_object, Bucket=dest_bucket, Key=file_name.split('.')[0] + '.jpeg')
30                 print("File has been converted to '.jpeg' format")
31         else:
32             return "File is not an image"
33     except Exception as e:
34         print(e)
35         print("Error getting object {} from bucket {}. Make sure they exist and your bucket is in the same region as this function.".format(file_name, bucket))
36         raise
```

Figure 40 code for lambda function

The code above is simple to understand, it does the following:

1. A lambda function will trigger on the source bucket when a file is uploaded to the source bucket.
2. Read all the images files in source bucket.
3. Convert all the "PNG" files into "JPEG" and vice versa.
4. Ignore files other than "PNG" and "JPEG", if any.

6.3.6 Testing our Lambda function

We upload some files into our source bucket ("fypsourcebucket"), which include some images and a text file, as shown in the below figure:

Destination s3://fypsourcebucket		Succeeded 6 files, 525.8 KB (100.00%)	Failed 0 files, 0 B (0%)
Files and folders			
Files and folders (6 Total, 525.8 KB)			
Find by name			
Name	Folder	Type	Status
2.JPG	-	image/jpeg	Succeeded
3 percent alert.txt	-	text/plain	Succeeded
3.JPG	-	image/jpeg	Succeeded
4.JPG	-	image/jpeg	Succeeded
5.JPG	-	image/jpeg	Succeeded
6.JPG	-	image/jpeg	Succeeded

Figure 41 Files uploading

Now, after uploading files to the S3 source bucket ("fypsourcebucket"), the lambda function will trigger on the source bucket and will do the operation that we defined in Python in lambda code. It should convert all the "PNG" files into "JPEG" and vice versa, and store them in the destination bucket ("fypdestbucket"), and if there is a file other than an image format, then it should be ignored.

6.3.7 Results

Amazon S3 > Buckets > fypdestbucket	
fypdestbucket	
Objects Properties Permissions Metrics Management Access Points	
Objects (5)	
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more	
Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload	
Find objects by prefix	
Name	Type
2.png	png
3.png	png
4.png	png
5.png	png
6.png	png

Figure 42 Result of lambda function

Hence, it has been proved that the lambda function works perfectly. Therefore, if we want to do any operations on files for the eBike in the future, then we just need to change the code of the lambda function, and the remaining process will be the same. This is the end of the cloud portion.

7 CONCLUSIONS AND FUTURE WORK

7.1 Conclusion

This chapter presents a conclusion to the work described in this thesis. The main contribution is summarized, followed by some suggestion regarding future direction of research related to this work.

7.1 .1 Important Finding

The following are important findings from this research project.

- Data acquisition system is designed that get data from the Electric bike.
- A precise current detection circuit is designed with Arduino microcontroller.
- A GPS sensor is interfaced which gives location of thievery
- A SD card module is used to store all the data.
- Some tests are done on SD Card to check the capability of SD Card
- Data transfer to SD card, ADC time, and Timer ISR are calculated
- Operation on data over the cloud

7.1.2 Future Recommendation

A number of problems were discovered throughout the research for this thesis that might benefit from further examination.

Many gaps in knowledge were discovered throughout the literature review. While the study provided in this thesis has addressed some individuals, others have not.

The following recommendation must be kept in mind for further work on this design system in future.

1. Design a machine learning module for analysis.
2. Write program for lambda function to do analysis over the cloud.
3. Develop a mobile application to send data direct to the cloud instead of uploading from local drive.

REFERENCES

- [1] A. Ulinuha, Jatmiko, and A. G. Riza, "Design and Implementation of Data Acquisition Device and Instrumentation Based on Microcontroller for Electric Motorcycle," *Proc. 2nd Borobudur Int. Symp. Sci. Technol. (BIS-STE 2020)*, vol. 203, Aug. 2021, doi: 10.2991/AER.K.210810.004.
- [2] S. Lekshmi and P. S. L. Priya, "Range Extension of Electric Vehicles with Independently Driven Front and Rear PMSM Drives by Optimal Driving and Braking Torque Distribution," *2020 IEEE Int. Conf. Power Electron. Smart Grid Renew. Energy, PESGRE 2020*, pp. 1–6, 2020, doi: 10.1109/PESGRE45664.2020.9070246.
- [3] X. Sun, Z. Shi, Y. Cai, G. Lei, Y. Guo, and J. Zhu, "Driving-Cycle-Oriented Design Optimization of a Permanent Magnet Hub Motor Drive System for a Four-Wheel Drive Electric Vehicle," *IEEE Trans. Transp. Electrification*, vol. 6, no. 3, pp. 1115–1125, 2020, doi: 10.1109/TTE.2020.3009396.
- [4] L. Tian, L. Wu, X. Huang, and Y. Fang, "Análisis paramétrico de la autonomía de los vehículos eléctricos propulsados por motores interiores de imanes permanentes considerando los ciclos de conducción," *CES Trans. Electr. Mach. Syst.*, vol. 3, no. 4, pp. 377–381, 2019, [Online]. Available: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8945550>
- [5] "No Title," 2018. https://www.researchgate.net/publication/354056638_Design_and_Implementation_of_Data_Acquisition_Device_and_Instrumentation_Based_on_Microcontroller_for_Electric_Motorcycle
- [6] V. Schwarzer and R. Ghorbani, "Drive cycle generation for design optimization of electric vehicles," *IEEE Trans. Veh. Technol.*, vol. 62, no. 1, pp. 89–97, 2013, doi: 10.1109/TVT.2012.2219889.
- [7] Y. Li, H. He, and J. Peng, "An Adaptive Online Prediction Method with Variable Prediction Horizon for Future Driving Cycle of the Vehicle,"

- IEEE Access*, vol. 6, pp. 33062–33075, 2018, doi: 10.1109/ACCESS.2018.2840536.
- [8] D. McDonald, “Ac 2010-1026: Data Acquisition in a Vehicle Instrumentation,” 2010.
- [9] S. Qiang and L. Chenguang, “Data Acquisition System for Electric Vehicle’s Driving Motor Test Bench Based on VC++,” *Phys. Procedia*, vol. 33, pp. 1725–1731, 2012, doi: 10.1016/j.phpro.2012.05.277.
- [10] M. E. Hairr, P. Griffith, J. R. Bailey, and W. Madden, “Data Acquisition System for Electric- and Hybrid-Electric Buses,” vol. 3, pp. 413–419, 2009.