CAR TO CAR COMMUNICATION

Final report

ABSTRACT

Car to car communication is a project done by Manzi and Ashraf, it was inspired by the Truck Platooning which is a new technology which gives one car control over all the car, the lead car takes control and the other car follows what the lead car does..

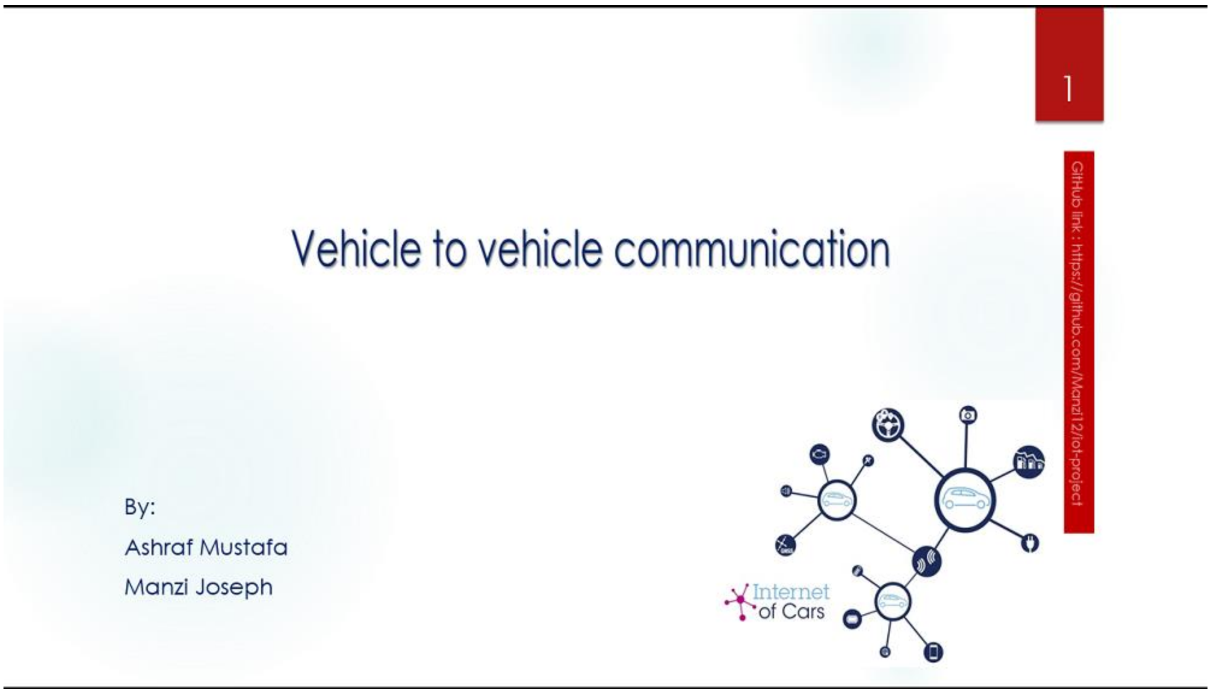
20079460

Applied Computing in the Internet of Things

TABLE OF CONTENTS

INTRODUCTION

This is a project carried done in our first year of college, it’s not perfect but we achieved our goal which was the main important thing and we learned a lot from the mistakes that we did and software and hardware problems that we encounter on our way to this amazing thing. This document will contain all the information that you need to know about the project and the source code, problems met during the creation of this project and all other important thing as listed below, and the presentation flash card will also be a guide to what the text will be talking about



1

Vehicle to vehicle communication

By:
Ashraf Mustafa
Manzi Joseph

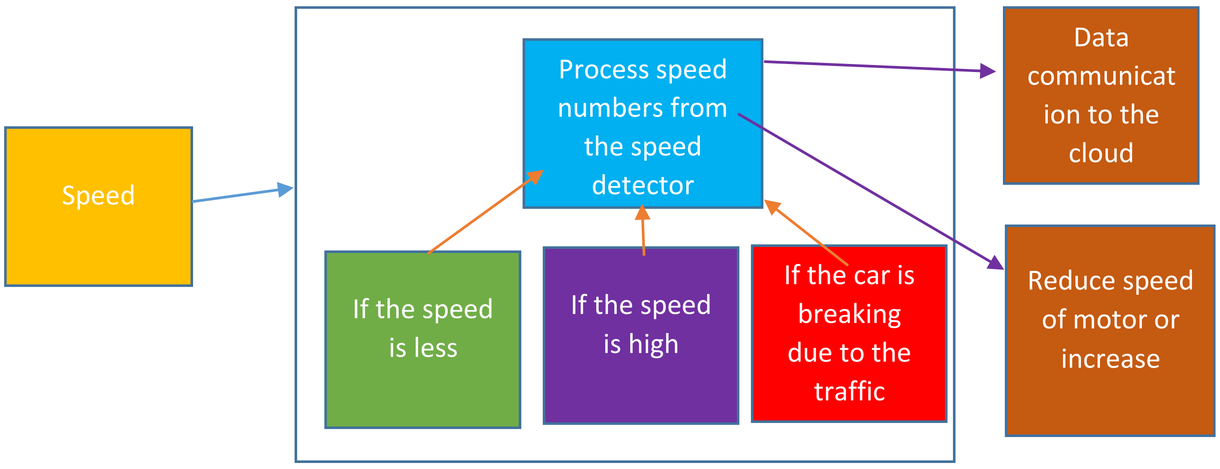
GitHub link : https://github.com/Manzi12/iot-project

Internet of Cars

First year in college we had no idea what to do for our internet of things project,my first idea was to create a security cctv camera,but it neeed a lot of things which I couldn’t do with the amoiunt of time that I had,so I decided to do the car racing with another friend who had already chose the car project.then the lecture decided that we could do the car to car communiatin to allow the two car to communicate through a cloud,one car should be able to send the speed data to the clould and the other car should be able to request the spped value and use them,and aprt from that the lead car should be able to be controlled on the cloud as well e.g giving it a speed through the cloud and then it controls the othe.after few days of consideration we though why not lets fo the car to car communocation..thats how all began.

FUCTIONAL DECOMPOSITION

input **processing** **output**


Speed
Process speed numbers from the speed detector
If the speed is less
If the speed is high
If the car is breaking due to the traffic
Data communication to the cloud
Reduce speed of motor or increase

Description of the subsystem.

Name: if the speed is less

Description: well this subsystem reports when the speed of the car is less than expected.

Input: the input for this is speed.

Output: the output is the speed increase in the motors and communication of the data

Processing: the processor manager will increase the speed motor and get the car back on track

Name: if the speed is high

Description: well this subsystem reports when the speed of the car is high than expected.

Input: the input for this is speed.

Output: the output is the speed decrease in the motors and communication of the data

Processing: the processor manager will decrease the speed motor and get the car back on track

Name: if the car is on the traffic lights

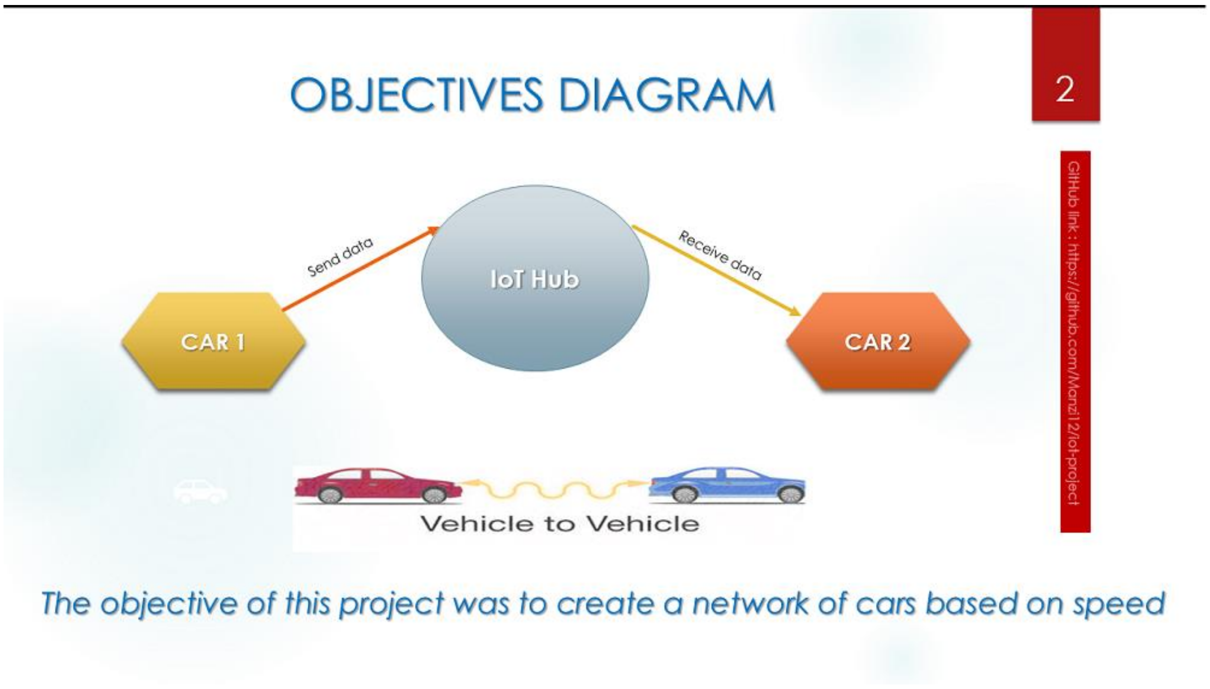
Description: well this subsystem reports when the speed of the car is less than expected.

Input: the input for this is light detector (hasn't been implemented yet).

Output: increase the speed gradually when the lights turn green

Processing: the processor manager will wait for the light to turn green and then then increase speed gradually until we get back to normal speed.

OBJECTIVES DIAGRAM



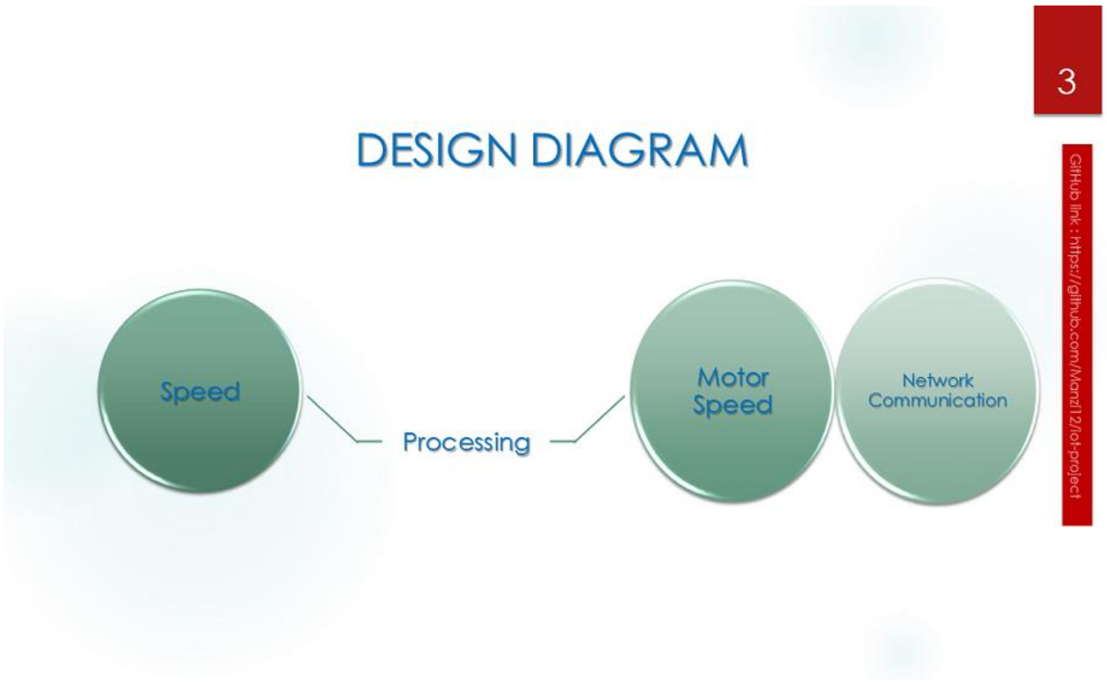
GitHub link : https://github.com/Manzi12/iot-project

The objective of this project was to create a network of cars based on speed

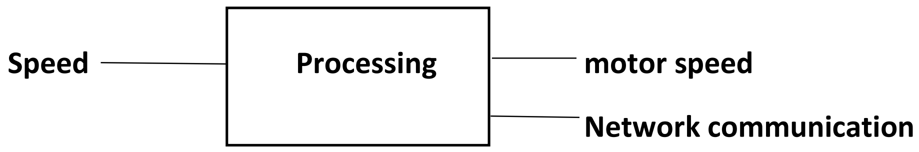
Like I said in the introduction that the objectives of this project was create car to car communication and as you can see from the diagram above that there is an IOT Hub which is the cloud which receives the speed values for the car and controls the speed as well and the second car get the speed from the cloud too, the IOt cloud that we used was adafruit io.and below is the link for the GitHub source code for the libraries that you need to set up the count and set the communication running

https://github.com/adafruit/io-client-python/tree/master/examples

DESIGN DIAGRAM

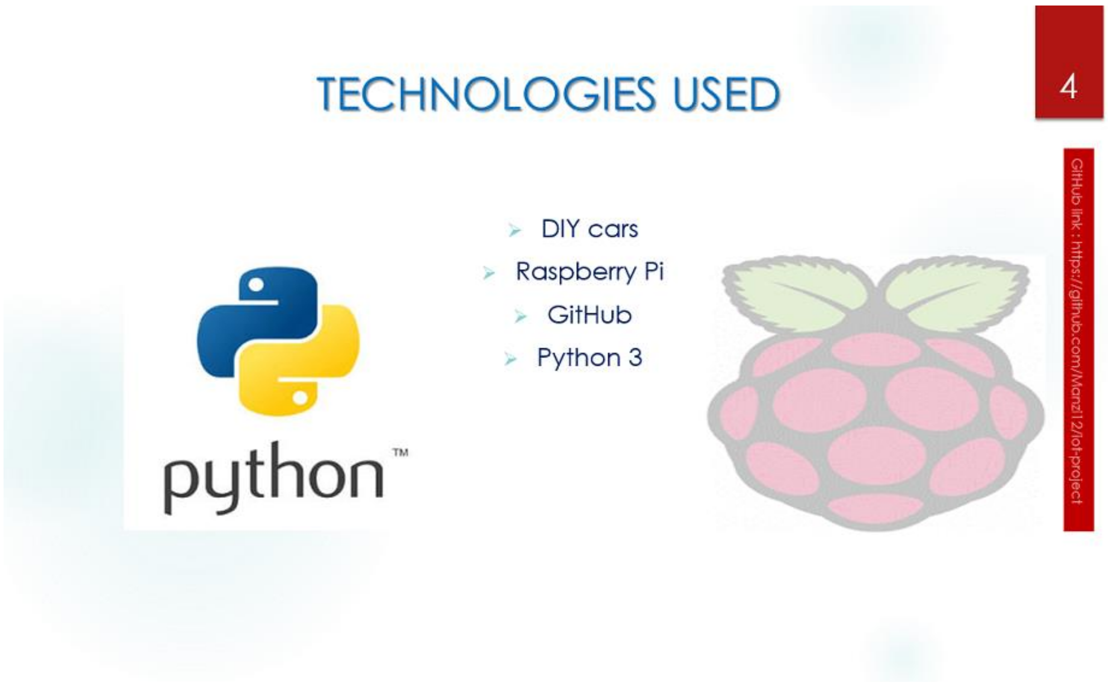


Context Diagram:



The design diagram shows how the inputs and the outputs of this car to car communication and as you can see from the diagram the input is the speed which is comes in from the leading car and the dashboard on the IOT hub adafruit, where we can also control the speed and this speed is processed and then sent to the motor speed of the two cars and its sent over using the mqtt protocol which plays a huge communication law between the two cars and the IOT hub.

TECHNOLOGIES USED



TECHNOLOGIES USED

4

- DIY cars
- Raspberry Pi
- GitHub
- Python 3

python™
GitHub link : https://github.com/Manzi12/iot-project

Well I have already tackled on some technologies that we have used in this project, I will just repeat then again in more details for you to have them in one place.

DIY cars

We used some DIY cars which were able to put in some motors to turn them into real cars not toys

Raspberry pi

This was the brain of the whole project we had two raspberry pi one for each car and the source code was inside, and it was actually setup to start once the pi was powered on and booting.

Mqtt

This was the communication protocol used to allow the communication between the cars and the iot hub adafruit

Adafruit

This was the IOT server / hub used to take in and analysis the data and pass it on to the other car, and we also had a dashboard on it which we used to control the speed of the leading car.

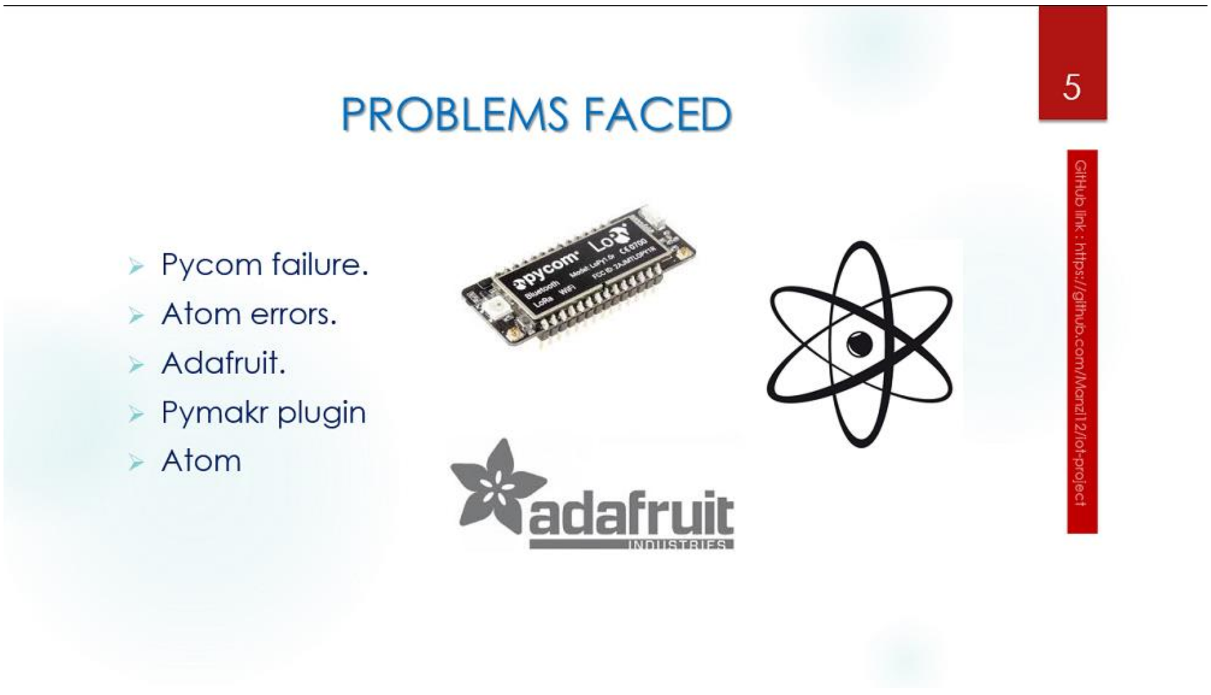
GitHub

This is a platform that we used to manage our project code, we have on it version which didn't work and version which worked so check it out the ink is down below

Python 3

This is the IDE used to code the python code for the cars

https://github.com/Manzi12/iot-project



5

PROBLEMS FACED

- Pycom failure.
- Atom errors.
- Adafruit.
- Pymakr plugin
- Atom

GitHub link : https://github.com/Manzi12/iot-project

pycom
LoPy
adafruit
INDUSTRIES

Life without problems is not fun, and it's quite impossible to manage to accomplish something great without going through some hard times and stressing moments and this project, it was not different at all we met lots of problems, so I will just break them one by one and give few details about them

Pycom failure, atom and pymakr plugin

So in the beginning we intended to use sigfox chip set because they were small and they had lots of functionalities in them e.g. they had accelometer, pressure and other important sensor which we could have tried to use to have lots of functionalities to our cars, but unfortunately their IDE which is called atom could not allow us to use the pymakr plugin which supports the use of the sigfox chips.so we wasted our time trying to get it to work but then it was officially declared having problems so we dropped it and searched for other options and that's when we switched to the raspberry pi.

SOURCE CODE

Reciever.py

The source code below is the one that goes in the car that is receiving the data form the cloud

```
//////////////code starts here ///////////
# Import library and create instance of REST client.
from Adafruit_IO import Client
from time import sleep
aio = Client('adafruit api key')

import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(13,GPIO.OUT)
my_pwm=GPIO.PWM (13, 1000)

# Get the last value of the temperature feed.


# Print the value and a message if it's over 100.  Notice that the value is
# converted from string to int because it always comes back as a string from IO.


#print('speed: {0}'.format(speed))

my_pwm.start(60)
while True:
    data = aio.receive('speed')
    speed = int(data.value)
    my_pwm.ChangeDutyCycle(speed)
    speed = data
    print('speed: {0}'.format(speed))

//////////code ends here //////////////
```

Send.py

This is the code that goes into the raspberry pi that sends data to the cloud or receives the data from adafruit

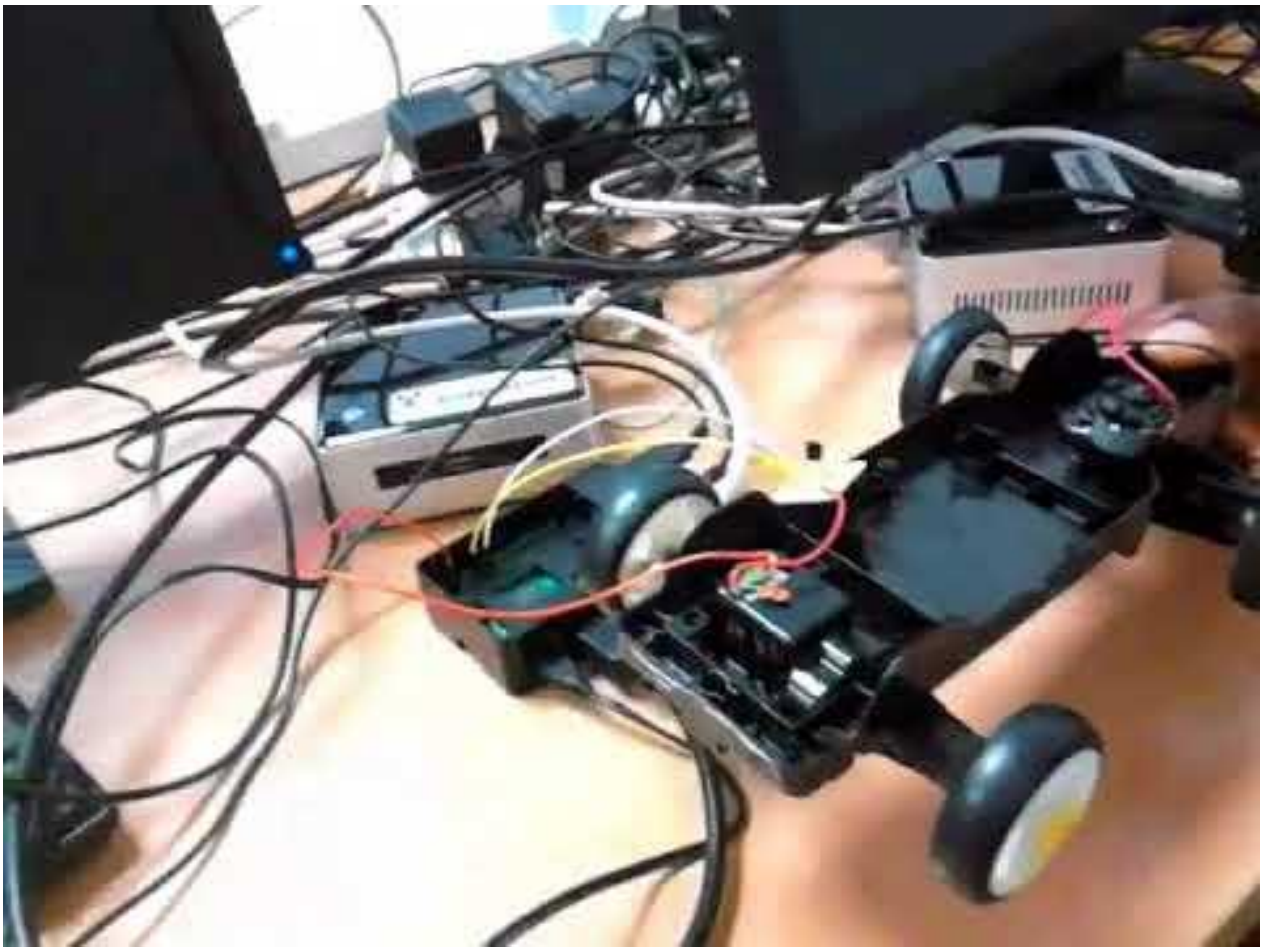
//////////code starts here /////////////

```
# Import library and create instance of REST client.
from Adafruit_IO import Client
from time import sleep
aio = Client('adafruit key')

import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(13,GPIO.OUT)
my_pwm=GPIO.PWM (13, 1000)

# Add the value 70 to the feed 'Speed'.
my_pwm.start(60)
while True:
   for i in range(70,100):
      my_pwm.ChangeDutyCycle(i)
      sleep(1)
      speed = i
      print('speed: {0}'.format(speed))
      aio.send('speed', i)
```

//////////code ends here /////////////

IMPLEMENTATION AND TESTING

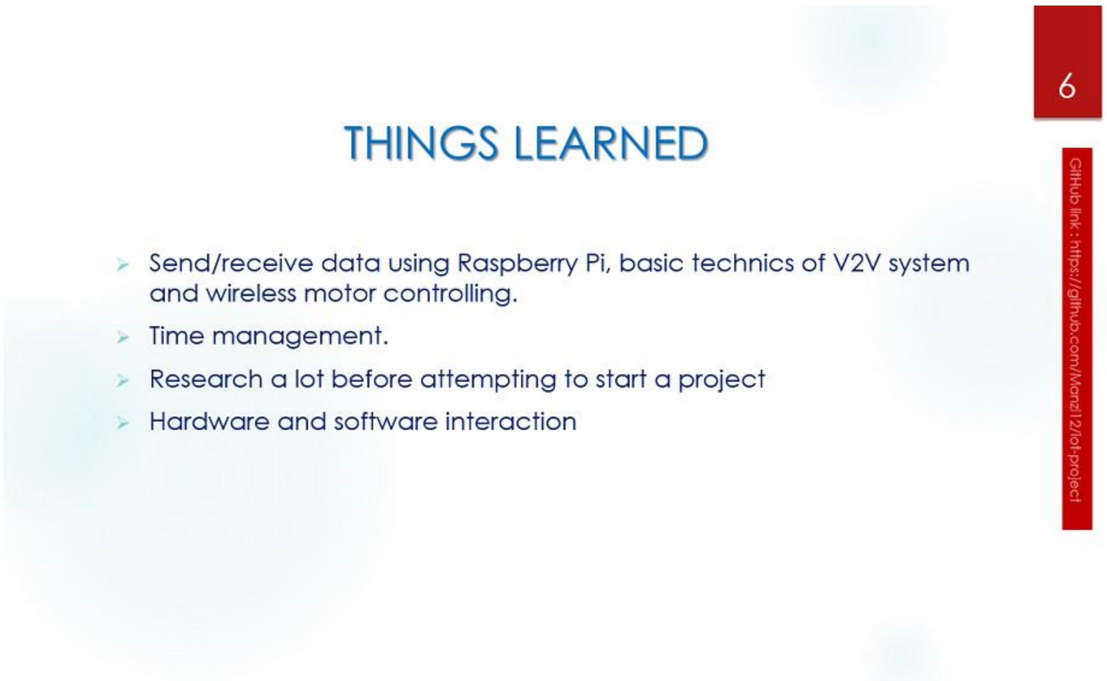
The implementation and testing were all done frequently, every time we added a n new code or improved on something or fixed a bug, the testing was done simultaneously to check if everything was in order, and below is the video I happen to take for some testing.



Below is the YouTube link for the video

https://youtu.be/Sxw1Cfw000A

THINGS LEARNED


6

GitHub link : https://github.com/Manzi12/Iot-project

THINGS LEARNED

- Send/receive data using Raspberry Pi, basic technics of V2V system and wireless motor controlling.
- Time management.
- Research a lot before attempting to start a project
- Hardware and software interaction

well given the problems that we met and the time that we wasted while trying to still debug the problems, it was really important to still manage to have something to show given the time that we wasted, so I would still say that we managed to manage time and it's something that we leaned.

Research

The other thing I would say we learned Is the power of researching before jumping into the whole project because at the end of the day you get stuck and then you start researching and end up finding out that what you did is wrong or its even hard you can't finish, so the research was really important to us.

Hardware and software interaction

Hardware and software interaction was one of the other thing I personally enjoyed about this given that I always wonder how hardware and software work together, so this time around I was behind the hardware and software to put it all together and see how to they all interact and accomplish the goal you initially set.

User Manual

well to do the same project its really easy just follow the steps below for a guide

1. You need the following equipment's first
    - the cars,
    - 2 raspberry pi
    - account on adafruit
    - connecting wire
2. Then just take the source code in the slides and put them in the raspberry pi for the two cars, make sure to put the API key of your account in the code so that your data should be sent to the right account.
3. Test first car and see if its working and then do the same for the second car, if all works out, then connect the raspberry pi on top of the cars and then try using the charged batteries to see if it all works like the way it was working before switching
4. After this steps if all goes well your cars should be racing each other, if they are not you probably made a mistake somewhere, so I would suggest to just go back each step and check them out

Well the user can really get lots of information and even more on our website were everything will be explained really well and in details, but you can also refer to the short video up the slides for how to control the car on the adafruit dashboard online.