Park-Smart with LoRa

Tejaswi Raj
MS, Department of Computer Engineering
Santa Clara University,
Traj@scu.edu

Ishit Sanghvi
MS, Department of Computer Engineering
Santa Clara University,
isanghvi@scu.edu

Sai Charan Medishetty
MS, Department of Computer Engineering
Santa Clara University,
smedishetty@scu.edu

Abstract—
This project presents the development of a cost effective, easy to use smart parking product based on embedded IoT and providing end to end solution using the cutting edge LoRa technology. It portrays the innovative methods developed to monitor data using the LoRa LPWAN. The parking system we propose is implemented using Esp32 HiLetgo LoRa transmitter, receiver and infrared sensors. LoRa receiver can receive the data from multiple transmitters as packets. The LoRa transmitter is installed in various parking lots to transmit the data of parking availability so that the use WiFi at every parking lot can be avoided. The nodes are deployed at multiple parking lots with one receiver at the control center which is connected to the cloud. The real time data can be monitored over the cloud at user's convenience on an android application/web-page deployed over AWS-EC2.

Keywords: IoT, LoRa, ESP-32, smart parking, real time data, AWS, Cloud

I. INTRODUCTION

The increasing number of vehicles on the road along with the mismanagement of available parking space leads to the parking related problems as well as increased traffic congestion in urban areas. Thus, it is highly required to develop an automated smart parking management system that would help the driver to find out some suitable parking space for his/her vehicle very quickly.
Parking has been a major issue all over the world for many people over many years recently which has led to the trend of smart parking
These are the overall objectives: define a decision support system for the smart parking management, to optimize returns on inputs, ensuring that resources are protected by utilizing technological capabilities such as data collection and analysis, accessed through a variety of devices: cellular, computer and all types of remote control and IoT.

A. *Communication Requirements*

- *Wireless device which can send data over long range i.e. more than 2 miles*. The wireless device is going to be used in the large parking spots where having Wi-Fi data relay is not a good infrastructural option. The device needs to communicate directly with the cloud to upload the gathered data.
- *Device which requires less power*. LoRa LPWAN consumes significantly less power as compared to Wi-Fi or cellular network
- *Low Data Rate*. The data acquisition and the related uplink data rate is quite small. The sensor data can be locally processed as part of the data aggregation and the transmission can be cryptic to minimize the bandwidth.
- *Large number of nodes to a single gateway* (approx. 1k/gateway using LoRa network). As part of minimizing the infrastructure cost, a single gateway is expected to handle a large number of data links, and support the data management in an efficient way. Various parking spots would transmit data to the LoRa RX

Since LoRa meets all these requirements it is incorporated in the communication system. LoRa transceiver HiLetgo **LoRa ESP-32 module with 1276sx modem** is used.

II. *OBJECTIVE*

A. *Purpose*

One of the significant issues that we are looking in the present society is traffic congestion mainly because of the unavailability of sufficient parking spaces at a given time. This has become a major problem in urban cities where there are metro stations, shopping malls, cinema theatres, IT parks, workspaces etc. We plan to develop a unique prototype which makes uses of embedded IoT, cloud and web to make a robust product.

B. *Background*

1. *Rational for the hardware and software architecture*

ESP32 is highly-integrated with in-built antenna switches, RF, power amplifier, low-noise receive amplifier, filters, and power management modules. We have chosen Hiletgo Esp32 module integrated with LoRa Tx/Rx. Esp32 has ultra low power features and is suitable for IoT applications

Our smart parking system consists of the following:

- Data acquisition through the IR sensors and display on LCD
- Data aggregation and initial processing in the MCU
- Uplink Transmission of the processed data over network received through another MCU
- Cloud based repository to store the data
- Downlink access through web/mobile application

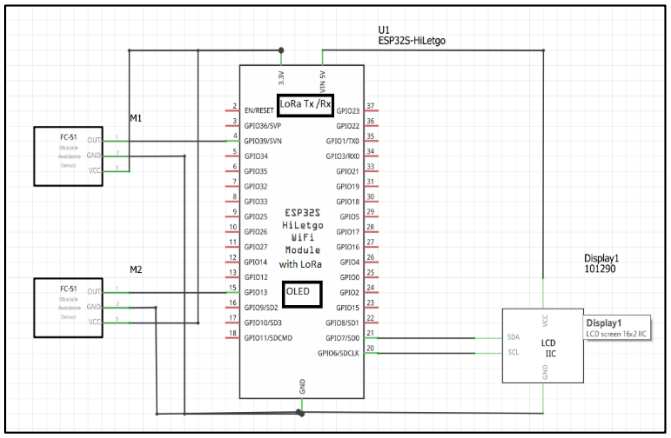

Fig 1. Hardware schematic done using Fritzing

2. Related Research To solve the problem

Due to the heavy traffic congestion in major metropolitan cities of California, the state has already adopted IoT based smart parking solutions in cities like San Francisco, LA, San Diego etc.

Companies like Semtech, Comcast, Sigfox are already establishing their IoT products throughout US.

Since WiFi based IoT solutions aren't feasible for all scenarios, LoRa based solutions are gaining huge prominence.

3. Related Products

The smart farming system has the following features:

- Comcast's MachineQ
- Smart Cities using LoRa by SemTech
- Place Pod by PNI sensor.

III. DESIGN

A. *Our Solution*

The MCU chosen for this work is the ESP-32. This meets all the data aggregation requirements and also has the options for power optimization which is an important consideration for this activity.

Initially we worked with LoRa Reyax 890 modules as TX, RX, however they were not integrating with the ESP-32 module. Therefore we shifted to HILetgo ESP-32 LoRa with OLED which met our requirements and functioned as expected. LoRa TX sends data in serial bursts and we had to segregate the data to upload it the cloud using some string manipulation techniques. The LoRa RX uploads data to Google Firebase (cloud)

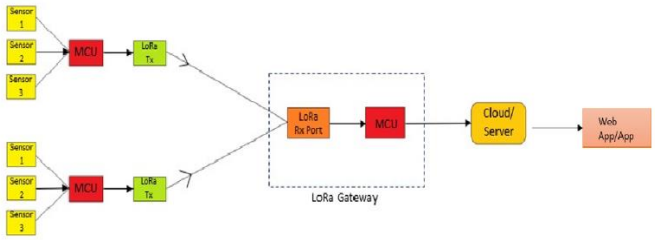

Fig 2. A representation of the transmission of data using LoRa WAN.

B. *LoRa Implementation*

We have used the communication system (LoRa transceiver) **LoRa module with 1278sx modem** in this smart parking system.

- On the Transmitter side, LoRa is integrated with ESP32
- On the Receiver side, LoRa is integrated with ESP32
- LoRa.h library is used to set the different parameters in ESP32 LoRa module
- LoRa spreading factor ranges from 7 to 12. With the increase in spreading factor; range increases while data rate decreases. As per our requirement for a moderate range and a data rate up to 5kbps, the Spreading factor is set to 12 (default).
- LoRa operates at 915E6 Hz frequency in US

C. *Hardware Components*

- 2 x HiLetgo ESP32 LoRa
- 2 x Obstacle avoidance IR Infrared Sensor Module EK1254x5
- 2 x 16x2 1602 LCD Arduino Display Screen Blue + IIC I2C Module Interface Adapter

D. *Cloud Storage*

Real Time data logs are stored in Google Firebase. The cloud acts as a repository for the data logs and the real time data from the IR sensors. The Firebase Real-time Database is a cloud-hosted NoSQL database that lets the

user store and sync data between multiple users in real-time.

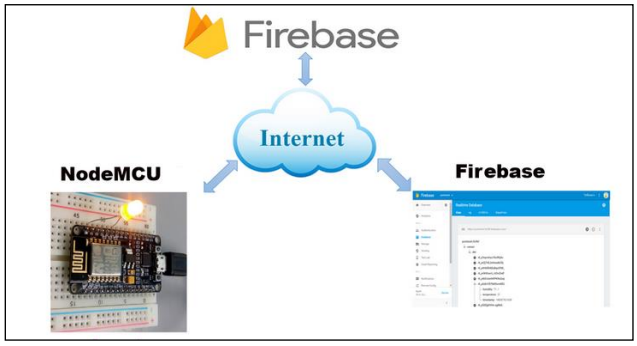

Fig 3. A representation of the cloud storage.

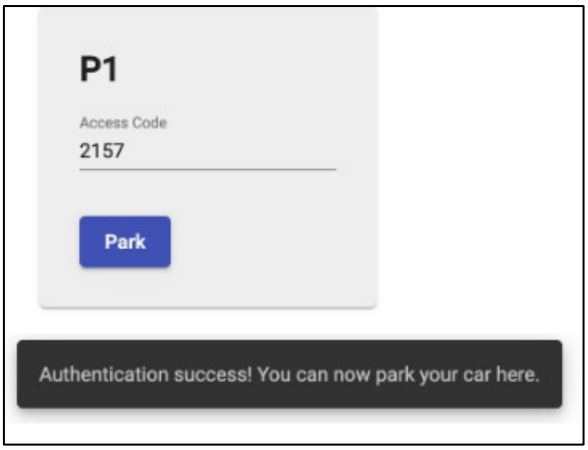

Fig 4. Authentication

E. *Software Architecture*

1. Firmware design and Implementation

The IR sensor has a digital output that's give a high level (1) when it detects something otherwise it's low (0), the detection range can be set by the potentiometer on the module.

The LCD is wired to the ESP32 by following the schematic diagram. We're using the ESP32 default I2C pins. Before displaying text on the LCD, we had to find the LCD I2C address (which was 0x27).

I2C uses only two wires SDA and SCL to transmit data between devices. I2C combines the best features of SPI and UARTs. With I2C, we can connect multiple slaves to a single master and we can have multiple masters controlling single, or multiple slaves.

2. Frontend design and Implementation

Angular JS is used for front end development.

The web page is hosted on AWS-EC2. http://18.144.88.150/

Front-end is done through HTML and CSS

The user can reserve the parking spot at his convenience.

The user is given an access key for secure authentication of the parking spot.

The user can reserve the parking spot based on the status (i.e. vacant/booked/available).

As soon as the user books the spot, an access key is generated which will be used for secure authentication at our smart-parking model.

After the access key is generated, a timer of is set (10s for demonstration purpose) within which the used needs to enter the access key at the parking lot and this is done for security purpose.

The parking spot becomes vacant for other users after fixed time frame.

This model can be easily upscaled for future requirements in real time.

4. Backend design and Implementation

The webapp is desgined in Angular 9 and serves to expose an interface that allows visitors to reserve parkings for their vehicles in a dedicated parking lot. A booking must be confirmed at the time of parking by a unique access code generated at the time of booking. Another interface is exposed to the user at the parking area and in charge of this activity. Once a parking lot is booked, the user is allowed to as much leeway to leave his car in the parking assigned to him as a specifically set timeout (which in our case for demo purposes is 10s).

JSONS are extracted from the cloud through the Angular application and displayed on the web page hosted on EC-2 instance.

Angularfire library was used to extract JSONS to the webpage.

The component driven architecture in Angular makes way for a dynamic system in which the number of operable parkings can be directly set in the backend without having to make subsequent changes. "Angular Routing" took care of the interface that accepts the unique user access code at the parking lot. The open source node module "AngularFire" is responsible for communication of the webapp with "Firebase" real-time database that holds values representing the state of the parking in among - vacant, booked and occupied. 2-way binding in Angular has been exploited in building the UI that mirrors the status values in the back-end by binding certain values of the JSON snapshot from the database with the user-friendly fields on the front-end. This makes way for changes in the back-end to immediately update values in the webapp interface making it dynamic and fast. What makes an Angular app aesthetic is the "Angular Material" add-on integration which in our case has furnished snackbars, modal dialogs and material cards.

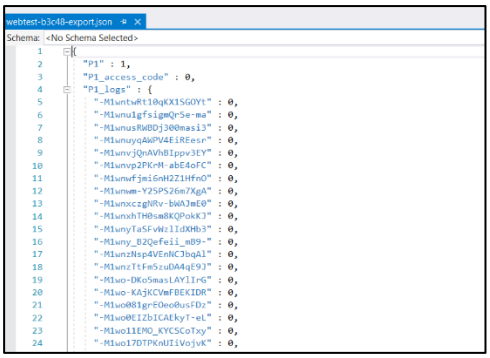

Fig 5. Database Schema

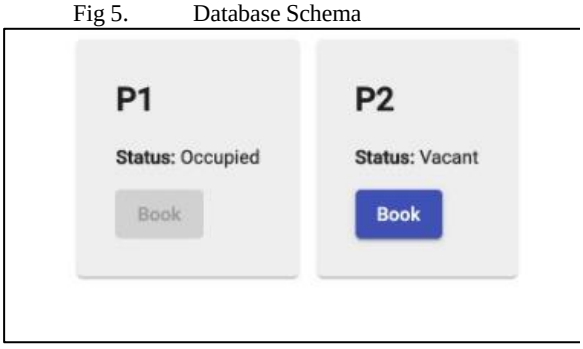

Fig 6. Vacant Parking Spot

5. *User Interface*

The user can check the status of the parking lot on the LCD or on the web-page mentioned before. The status of the spots are displayed in real-time. The user can book the spot at his convenience remotely.

The LCD is refreshed every second.

Phonegap-hybrid cross platform ap is also developed for the user to view/book the parking spot.

Fig 7. LCD Status

6. *User Testing*

The status of the spots are displayed in real-time. The user can book the spot at his convenience remotely through the webpage/app.

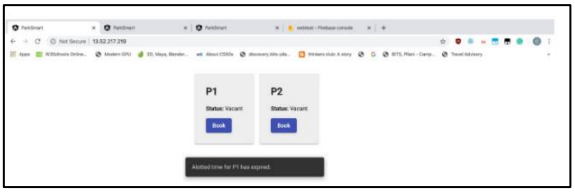

Fig 8. Web portal

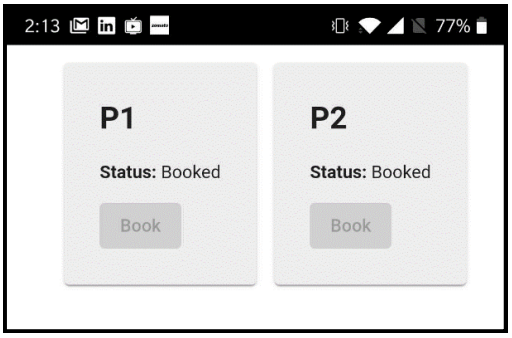

Fig 9. Phonegap app

IV. RESULT

1. We were successfully able to deploy our product and test it's functionalities.
2. LoRa was functioning with a long range!
3. The web page has a web authentication portal and displays the parking slots in real time

V. CONCLUSIONS

1. Using this smart parking system, one can easily check the availability of the parking slot using internet anywhere in the world.
2. This would benefit a lot of smart cities which are facing problems with parking issues.
3. Traffic congestion can be avoided.
4. The project can be easily upscaled for future purposes.

VI. LESSONS LEARNED

We had a hand's on experience on IoT and WoT.

We initially worked with LoRa REYAX 890 transceivers but the libraries were not compatible with ESP32.

We worked with HiletgoESP-32 LoRa module to resolve the issue. Power management of the sensors, modules were also inferred. We had issues with IR sensors, but we managed to resolve it with proper grounding. We had a great learning experience learning angular JS and designing the front end.

VII. REFERENCES

[1] Ravi Kishore Kodali, Krishna Yogi Borra, Sharan Sai G. N. and Jehova Honey Domma, "An IoT based Smart Parking System using LoRa", "2018 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery"

[2] M Farooq and D Pesch, "Analyzing LoRa: A use case perspective," 2018 E 4th World Forum on Internet of Things (WF-IoT).

[3] S Mendiratta, D Dey and D Rani Sona, "Automatic car parking system with visual indicator along with IoT," 2017 International conference on Microelectronic Devices, Circuits and Systems (ICMDCS).

[4] Jonathan de Carvalho Silva, Joel J. P. C. Rodrigues, Antonio M. Alberti1, Petar Solic, Andre L. L. Aquino, National Institute of Telecommunications (Inatel), Santa Rita do Sapuca´ı - MG,

Brazil,Instituto de Telecomunicac¸ ˜oes, Portugal, University of Fortaleza (UNIFOR), Fortaleza - CE, Brazil, University of Split, Split, Croatia, Computer Institute, Federal University of Alagoas, Macei´o - AL, Brazil. “LoRaWAN - A Low Power WAN Protocol for Internet of Things: a Review and Opportunities”

[5] https://github.com/osresearch/esp32-ttgo

VIII. Apendices

1. *Hardware Inventory*

1) HiLetgo ESP32 LoRa SX1276

Vendor: Amazon

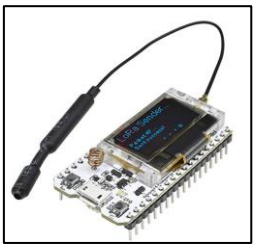

Fig 10. LoRa ESP-32

2) Gikfun Obstacle avoidance IR Infrared Sensor Module EK1254x5

Vendor: Amazon

Fig 11. IR sensor

3) 16x2 1602 LCD Arduino Display Screen Blue + IIC I2C Module Interface Adapter

Vendor: Amazon

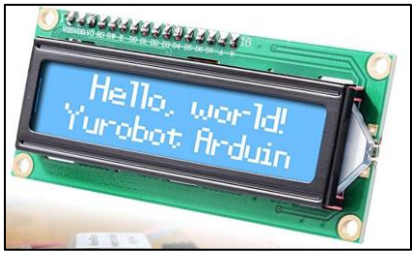

Fig 12. LCD

2. *Software Inventory*

1. *Web App*

Angular JS was used for the web app :
Front-end is done through HTML and CSS

2. *Hybrid App*

Phonegap was used to develop a hybrid app which works on IoS and Android

3. *Libraries*

The following libraries were used:

- SPI.h: Serial peripheral interface library
- LoRa.h: LoRa library

Libraries for OLED Display

- Wire.h: This library allows you to communicate with I2C / TWI devices
- Adafruit_GFX.h: Graphics library
- Adafruit_SSD1306.h: Adafruit_SSD1306.h
- WiFi.h: Helps to connect ESP-32 to internet
- FirebaseESP32.h: Helps to connect ESP-32 to Firebase

IX. Execution Instructions

1. *Hardware instructions*

Upload IR_LCD_LoRaTX.ino into the transmitter NodeMCU with IR sensors and LCD
Upload LoRa_Receiver.ino to the receiver NodeMCU.
The transmitter send the packets of data to the receiver.
The receiver segregates the data as per the code.
The Receiver is connected to WiFi. Change the SSD and password as required.
The receiver uploads the data into Firebase in real time.

2. *Web Application*

To start up our frontend and backend, all you need to do is SSH into the AWS EC2 instance and run ‘sudo service apache2 start’ This will run the Angular JS web application

X. Acknowledgement

We would like to sincerely thank Professor Amr Elkady, who inspired us, enhanced our knowledge and helped us push out limits with making this wonderful and successful project.