UNIVERSITY OF CALIFORNIA, SANTA CRUZ

Campus Parking Infrastructure: Team 3

Scott Birss, David Caplin, Troy Fine,
Chenxing Ji (Gabriel), Aditya Sriwasth, Nelson Yeap

February 16, 2018

CONTENTS

1 Project Objective

When it comes to parking on campus, it is safe to state that several improvements could be put in place to make it more accessible and less of a hassle for the user. The goal of this project is to improve the existing system in place by using the concept of IOT (Internet of things). By utilizing several hardware and software elements and tools, our group seeks to design and implement an affordable, scalable and robust solution to address this issue.

2 Project Summary

The project seeks to improve the user experience when it comes to parking in a parking lot infrastructure by designing a mobile application that will inform them about all of spots available at a certain time before they have to drive up to campus. The user will be able to select the parking lots they want to view and get updates about any parking lots opening up in real time.

At the parking lot, there will be sensors placed at every single space to determine if a spot is vacant or occupied. This sensor will be attached to a microcontroller that communicates with a cloud that acts as the central node for communication between all of the devices for this project. Once a user pulls over a vacant parking spot, the sensor will send a message to the microcontroller indicating that the spot is taken, and the user can use the bluetooth functionality on his/her phone to establish a connection with the microcontroller to authenticate themselves and signal that they intend to park at that spot. If the user is currently enrolled at UCSC, they can simply enter their Student ID to check if they are allowed to park at the spot, and proceed to leave the car once they get a notification on the phone indicating they are good to go. A guest would have the option to pay for parking also using their mobile device.

Finally, the parking administration would also have a website application to manage parking spots, set up reservations for special events, or perform other tasks remotely, thus making it an easier experience for not just the user but also the administrators of a parking infrastructure.

3 Design

This section describes the several components that will make up the design for the project. The elements required to design this project can be split up into two major categories: hardware and software.

3.1 Raspberry Pi

The Raspberry Pi handles not only handles the local micro-controllers as a central node, but also handles the communication between the hardware and the Cloud. Since the Raspberry

Pi doesn't have a sleep mode, the Raspberry Pi would be configured to disable most of its ports and remain at an idle state to maintain the state of minimum power consumption.

3.2 Cloud

The Cloud will serve as the central node for sending and receiving data to all of the components and establish an important communication link between all of the devices. The Cloud will conceptually have a front-end and back-end section, with the front-end designated for communicating with the user interface of the website and mobile application while the back-end designed to collect and process data coming from the Raspberry Pi(including the image captured from the camera) and the two applications. For this project, we are using the Google cloud platform.

3.3 Computer Vision

Integrated into the Raspberry Pi, the Computer Vision part of this project aims to convert the input image from a connected camera into a character string which is the license plate of the vehicle. This part incorporates with the current OpenCV library and simple CNN's aiming to have a higher accuracy on recognizing the license plate correctly.

3.4 Mobile Application

The mobile application is one that will be used by the client who wants to park at a spot. The application will give the user information about how many parking spots are open at a particular area, enable them to reserve a certain spot and also make payment in certain cases. We are currently planning to write the mobile application using the Apache Cordovta framework since this enables us to make it platform independent, enabling us to run it on an Android or iOs device.

3.5 Website Application

The web application will be an application that can be run on a browser. The application will give TAPS personnel unlimited access to the information on the cloud. This means they will be able to designate parking spots as taken even if they are not, essentially reserving them for people/events. We are currently looking into writing the webapp using the Apache Cordova software framework since this makes the API protocols much easier to handle since we plan on using the same framework to design the mobile application. Apache Cordova also relies on writing code in javascript, HTML and CSS, thus making it a seamless platform to utilize to design a web-based application. As of now we have determined that Cordova can establish communication with the cloud through the use of a plugin, but more research will have to be done to make sure that Cordova can be used for this project.

4 Hardware Parts List

Quantity	Item	Price
2	Raspberry Pi 3	$60
2	Raspberry Zero	$20
2	Adapters for Pi Zero	$30
2	PSoC CY8CKIT-049-42XX	$10
2	PSoC 5	$30
2	PAdafruit Pi Display	$10
2	LANDZO 7 Inch Touch Screen	$86
2	Pi Camera	$26
2	SR04	$10
Subtotal		$282

5 API List

The API list will specify the protocols we will use to establish communication between a device/application and the cloud. Our goal is to utilize several REST API's that will enable the mobile and web application to establish connection with the cloud.

5.1 Arduino ⇔ Raspberry Pi

Communication between the Arduino and Raspberry Pi will be done via GPIO pins. The Arduino has two output pins. The first pin is a parking status pin. An empty parking space is signified by a low signal and turns high when occupied. The second pin on the Arduino is a valid pin. This turns high when the state of the parking space changes. The valid signal will remain high in till it receives an acknowledgement signal from the Raspberry Pi (ack). This is to prevent the Raspberry Pi from missing data sent from the Arduino. In addition, the valid signal acts as a wake up signal for the Raspberry Pi. See figure 5.1 and 5.2 to see an example of the signals.

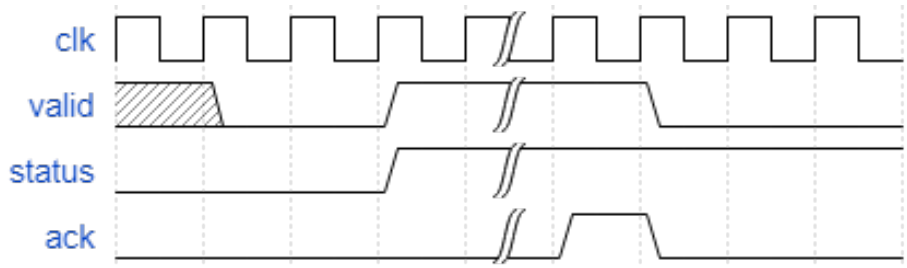

Figure 5.1: Car pulls into a spot

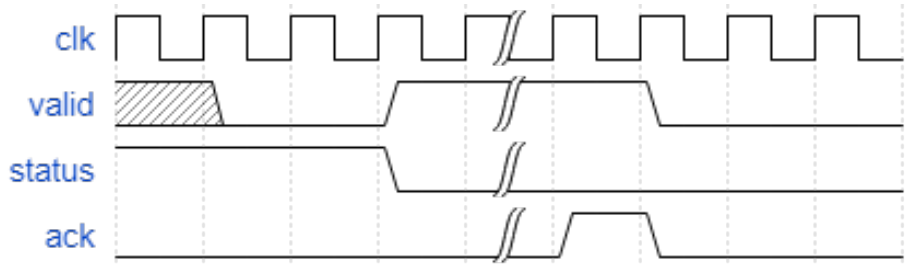

Figure 5.2: Car pulls out of a spot

5.2 Gateway ⇔ Cloud

When a car arrives in a parking space and the Raspberry Pi sends the space data to the cloud

- claim_spaces : space_num[ ] | code[ ]

A car leaves the parking space

- free_spaces : space_num[ ]

The Admin requests a picture from the space

- get_spot_pic : space_num[ ]

5.3 Administrator Web App ⇔ Cloud

Administrator wants to create a new parking lot

- create_lot : space_num[ ] | lot_name | permission_level | floors

Administrator wants to reserve a space

- reserve_space : lot_name | space_num[ ] | floor

Administrator wants to add a student to the database of students along with their corresponding parking permission level

- add_student : name | permission_level | code

Administrator wants to view the status of a lot

- view_lot_status : lot_name | floor

Administrator wants to erase an existing parking lot

- destroy_lot : lot_name

Administrator requests a pic from a parking space

- get_spot_pic : lot_name | space_num | floor

5.4 Mobile App ⇔ Cloud

For the mobile application, our goal is to design the first prototype of the app using Android studio since it enables us to utilize several predefined API's as they work well win the Google cloud platform. To authenticate a user to the cloud, we will utilize the API called "Interface authenticator". To carry out the following tasks listed below, we will utilize a task queue and set up a push and pull queue to dispatch requests to and from the server. The application that creates the task is not notified whether or not the task completed, or if it was successful. The task queue service provides a retry mechanism, so if a task fails it can be retried a finite number of times.

User wants to view the status of a lot

- view_lot_status : lot_name | floor

User wants to claim a parking space

- claim_spaces : space_num | code

5.5 Bluetooth Beacon ⇔ Mobile App

When a user arrives at a parking spot:

- User authenticate(HttpServletRequest request)