Automatic Parking Monitoring and Assistance for city of Champaign

ECE 445 Design Document

Christopher Santoso
Ximin Lin
Bo Wang
Group 17
TA: John Kan

1. Introduction

1.1 Objective

Parking spots on campus are numerous, and there is constantly a need for multiple people to constantly monitor parking spots to ensure rules are not broken. It is very easy for an individual to make an honest mistake however, such as parking too close to the curb, forgetting the time limit they paid for, or parking in a spot they are not allowed to park in.

We propose to solve this problem with an enhanced parking device that can be either mounted on a pole or a wall that will monitor cars coming in and out of the parking space, and notify the car-owner and the officials if there is a violation. Our device will also be able to assist the individual by giving signals in the form of color-coded lights on the device to assist them in parking (e.g. if they are too close to the curb or outside their spot, a visible light on the device will turn red. Stays green otherwise).

1.2 Background

When people forget to pay the parking meter due to carelessness or some emergencies, people could be charged for $50. Also, sometimes because of the bad schedule plannings, people ran late to the parking lot to pick up their car. In case of these scenarios, people usually pay extra money to safely cover the time they will use, which in turns cause people losing money because they left early. Our meter attempts to solve all these problems.

Our meter will check the person's balance on the server and calculate the time the person parks the car. Remote server will automatically charge from the account for the time the car parked. When balance in this person's account runs out, we will notify the person first. Through our website, they can pay the account. As a result, the person will only receive parking ticket when he or she refuses to pay. This could greatly reduce the waste of money and accommodate some emergencies.

1.3 High-level requirements list

• Able to recognize if there is a car parked in the space (via. ultrasonic sensors), which is defined as a car sitting at a distance of 0.35 meters +/- 0.25 meters,

• Able to identify its license plate within the next 1 minute of detecting a parked car, and associate the license plate with a user account (if one exists), and begin charging the driver's account after 5 minutes of being parked.

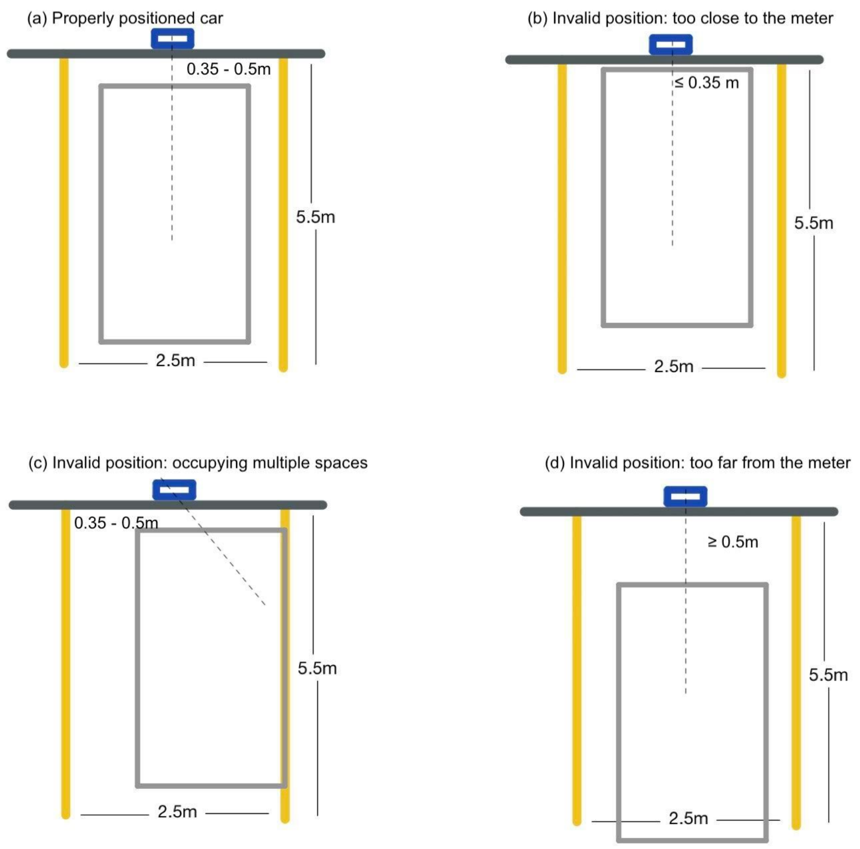
• Able to recognize if a car has committed a violation†, and give a physical indication (LED color) to the driver within 10 seconds of the violation being committed.

• Upon the car leaving the parking spot, the database should accurately reflect the parking time and updated balance within 5 minutes of the driver leaving. This change should be reflected on the web server (viewable by the driver) within 1 additional minute.

†A violation is defined as:

- *a car either being too close (12cm +/-3) or too far (46cm or more) from the parking meter.*
- *a car parked during restricted hours or in a reserved spot*
- *a car occupying multiple spaces*

2. Design


(a) Properly positioned car
0.35 - 0.5m
5.5m
2.5m
(b) Invalid position: too close to the meter
≤ 0.35 m
5.5m
2.5m
(c) Invalid position: occupying multiple spaces
0.35 - 0.5m
5.5m
2.5m
(d) Invalid position: too far from the meter
≥ 0.5m
5.5m
2.5m

<u>Figure 1:</u> Proper versus invalid parking positions. (a) A properly positioned car. License plate is aligned with the camera module (≤30 degrees offset from the center of the camera's center view). (b) Car is parked too close to the meter, which is less than 0.35 meters from the meter itself (in which case, "an Obstructed View" message will appear). (c) Car is occupying multiple spaces, not leaving space for other cars to park, which is measured by the license plate being ≥30 degrees offset from the center of the camera's view. (d). Car is parked too outwards, which is defined as the car being more than 0.5 meters away from the module.

2.1 Block Design

Top-Level Block Diagram

Figure 2 and *Figure 3* depict the connections and requirements of the PCB module and the Raspberry Pi/Microcontroller modules, respectively. Thus, the following section is divided into the Microcontroller, Ultrasonic Motion Sensor, and LCD Display module, and the Raspberry Pi and Microcontroller module.

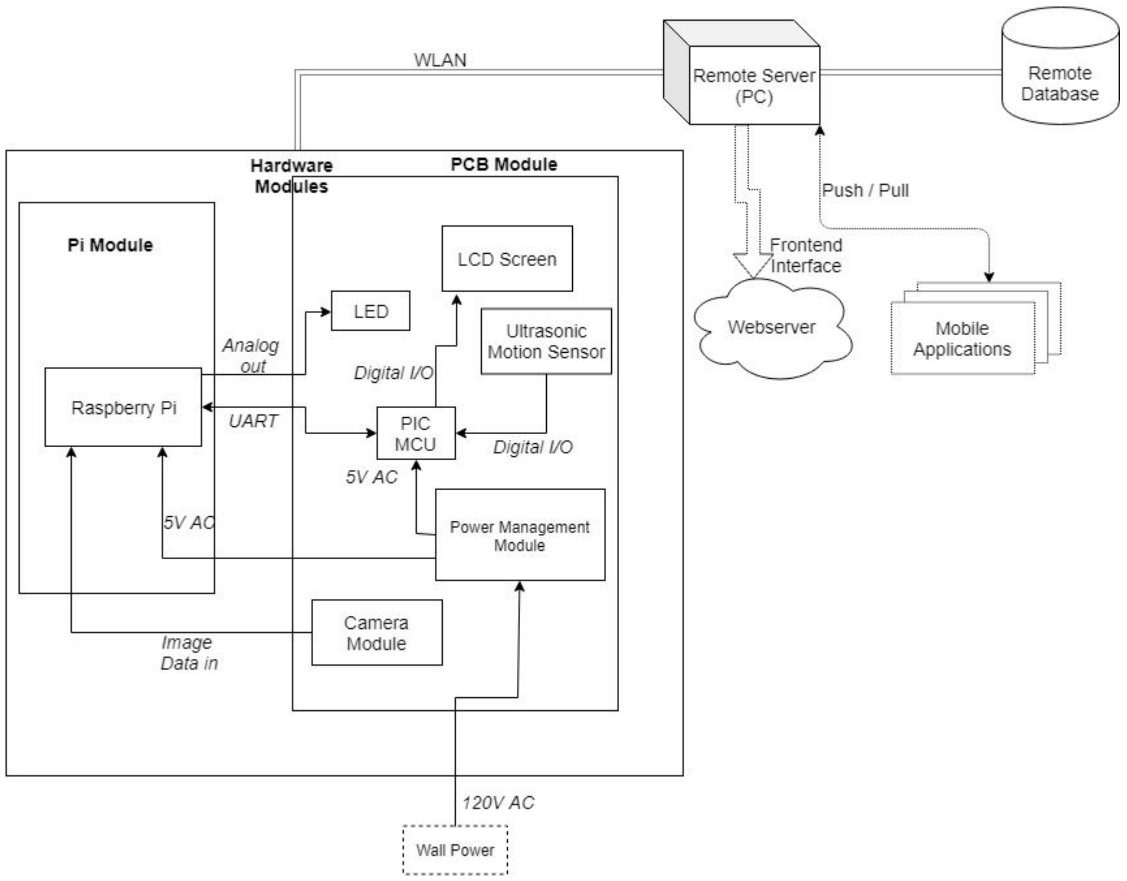
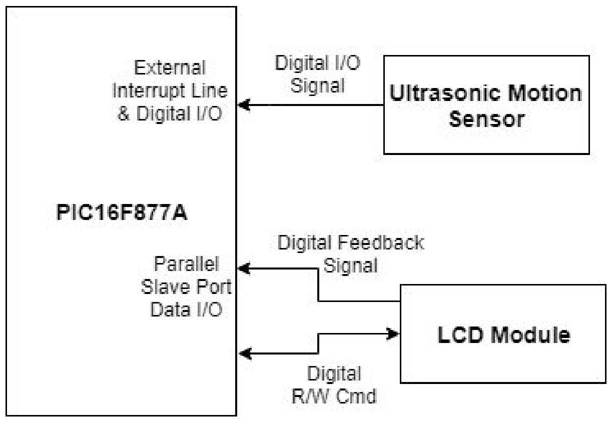


Figure 1: Top-Level block diagram of the entire system. The hardware section is separated into a PCB module, which will consist of the modules to be placed on the PCB, as well as the Pi Module, which consists of only the Raspberry Pi.

Requirement	Verification
The module must be able to communicate with the main server to send and retrieve license plate data.	A. Create a database entry with an associated predefined license plate on the server side. B. From the client, send a (debug) request to retrieve the plate entry and its associated balance. C. Verify that the client contains the same entry information as what was created in (A).
The parking module must be able to identify a license plate when there is one present within 0.5 to 1 meter, of the camera.	A. Position a license plate 0.5 meters from the module's camera. B. Check to see that the license plate number displayed on the LCD screen is the same as the license plate number held in front of the parking module.
The parking module must be able to fetch an associated balance from the server within 1 minute of identifying the license plate.	A. Position the plate until it is identified by the meter. B. Verify that the debug message on the LCD screen correctly displays the database entry associated with the licence plate.
The parking module must be able to detect a car leaving the parking space, which is defined as the car moving away more than 1 meter from the meter, and no identical license plate being identified within the next minute.	A. Position the license plate 0.5 meters from the parking module, and hold it in place for five minutes to signal the meter that a car is parked. B. Move the plate away from the module to a distance of 1 meter (or greater), and keep it out of range for two minutes. C. Verify that the LCD no longer displays a license plate or a balance.

2.2 Hardware (Microcontroller & PCB and Raspbery Pi Modules)

2.2.1 <u>Microcontroller, Ultrasonic Motion Sensor, and LCD Display</u>



<u>Figure 2:</u> PIC Microcontroller to LCD Module and Motion Sensor Module connections. The motion sensor will connect to the PIC's external interrupt line to notify the MCU of new incoming data, and send the data via. a digital signal to one of the PIC's digital I/O pins. The LCD module is commanded by the PIC via. another Digital I/O pin.

Requirement	Verification
The microcontroller must be able to display a 16x2 character message on the LCD screen.	A. Program the microcontroller to send a predefined 32-character ASCII message to the LCD display peripheral. B. Verify that the corresponding message appears on the display.
The microcontroller must be able to read a distance from the ultrasonic sensor.	A. Command the microcontroller to display the distance read from the ultrasonic sensor. B. Position an object in front of the ultrasonic sensor. Verify that the distance displayed changes in correspondence with the position of the object.
In displaying a status message (1. Empty Balance and 2. Obstructed Spot or Plate, 3. Restricted Hours), the LCD module will take no more than 3 seconds to display the correct status message upon the correct conditions being met.	<u>1. Empty Balance Status Test:</u> A. Load the parking module with an nearly-empty balance (0.40 USD). B. Allow the balance to empty by keeping a parked car in the same spot for more than 5 minutes. C. Verify that the LCD display reflects the

	"Empty Balance" message. 2. Obstructed Spot or Plate Test A. Place an object within 3 meters, 1 meter, and 1cm of the parking module with no license plate in view of the camera. B. Verify that for each of the 3 distances, the "Obstructed Parking Spot" message displays on the LCD. 3. Restricted Hours Test A. Set the meter's restricted hours to -1 and +1 hours of the current time. B. Place an object with a license plate within 0.5 meters of the parking meter's camera view. C. Verify that the LCD displays the "Restricted Parking Hours" message.

2.2.2 Microcontroller to Raspberry Pi Module

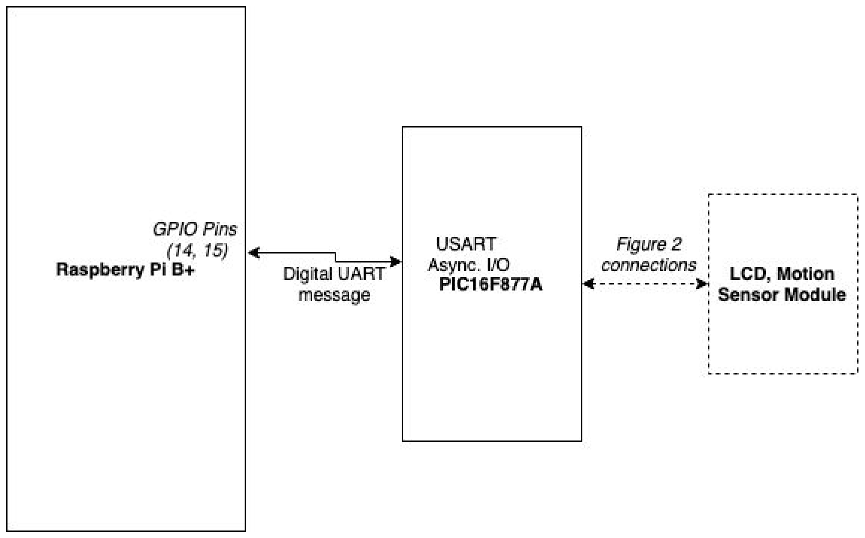


Figure 3. Raspberry Pi to Microcontroller connection. The dotted module is the module depicted in Figure 2 (connections between the microcontroller, the LCD screen, and the ultrasonic motion sensor).

Requirement	Verification
The microcontroller must be able to send and receive data over UART with a baud rate of 9.6kbaud.	A. Using PySerial, open a serial connection between the Raspberry Pi and the microcontroller (with 9.6kbaud transmission speed).

	B. Send a command to the microcontroller to give it a predefined message. C. Command the microcontroller to send back the predefined message. D. Verify that the message received is the same as the message sent.
The microcontroller must be able to wake the Raspberry Pi from sleep mode via.	A. Manually put the Raspberry Pi to into sleep mode by entering "sudo shutdown -h now" into the console. B. From the microcontroller, program it to manually send the wake up command upon a debug button (physical debug button) being pressed. C. Verify that the Raspberry Pi is able to wake up within 30 seconds of the button being pressed.
The Raspberry Pi must be able to send and receive a balance from the microcontroller, and the microcontroller must be able to update this balance (while a car is parked).	A. From the Raspberry Pi, send the command to update the microcontroller's stored license plate. B. From the Raspberry Pi, send the command to display the license plate on the LCD screen. C. Verify that the plate number displayed on the LCD screen is the same as the one sent.
The Raspberry Pi must be able to update the microcontroller with a new license plate (whenever a car is detected).	<u>Test 1</u> A. From the Raspberry Pi, send the microcontroller a preset license plate. B. Command the microcontroller to display the received license plate. Verify that it's the same as the one sent. <u>Test 2</u> A. Command the microcontroller to display the license plate number. B. Position the parking module such that a license plate is within its view. C. Verify that the license plate displayed on the LCD screen is the same as the plate in front of the camera.

2.3 Power Module

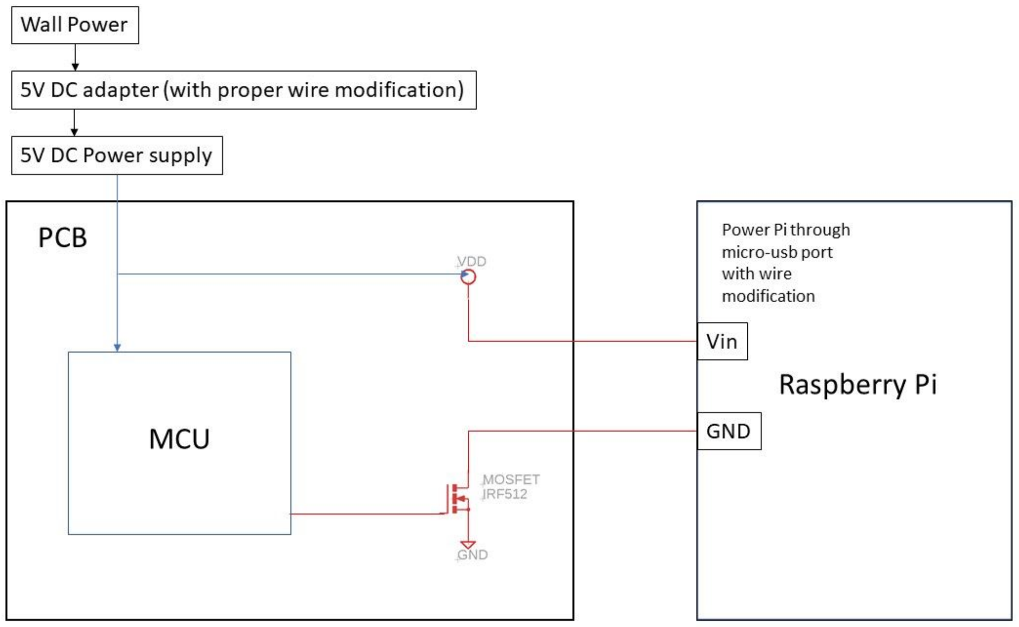

Wall Power
5V DC adapter (with proper wire modification)
5V DC Power supply
PCB
VDD
MCU
MOSFET
IRF512
GND
Power Pi through micro-usb port with wire modification
Vin
Raspberry Pi
GND

Figure 4. *To power our system, we first adapte the ground power with a 5V DC adapter, we then power our modules with 5V DC. We want our MCU to automatically wake up raspberry pi with high or low voltage. The circuit shown above works as a switch.*

Requirement	Verification
VCC should provide stable 5V and 500mA power for raspberry pi to safely work.	We will use oscilloscope to measure the voltage and the current when MCU allows VCC supplies raspberry pi.

2.4 Central Server

2.4.1 Top Level Software Flow

The central server will handle the main logic shown in the server-side software flowchart (*Figure 1*). Thus, it is responsible for managing the flow of data between the database and the main server computer, authentication and security (including user/admin sign-up), and transfer of information to and from the smart meters.

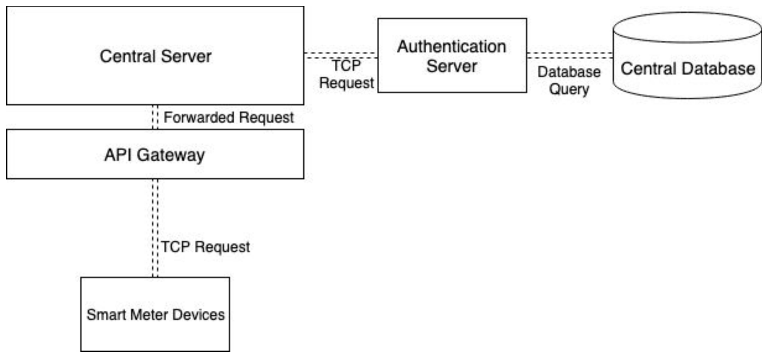


Figure 5: Top-level data flow of the entire system.

2.4.2 Server and MCU

The flow of data on the server-side is shown in the flowchart below. The server will be responsible for handling requests that come through the API gateway and making the correct query to the database to check if the license plate exists. If the plate is not already registered, a new entry will be created with a balance of 0, and when the driver with that license plate signs up, any remaining balance will be saved to their account, and their user information will be synced with their existing plate.

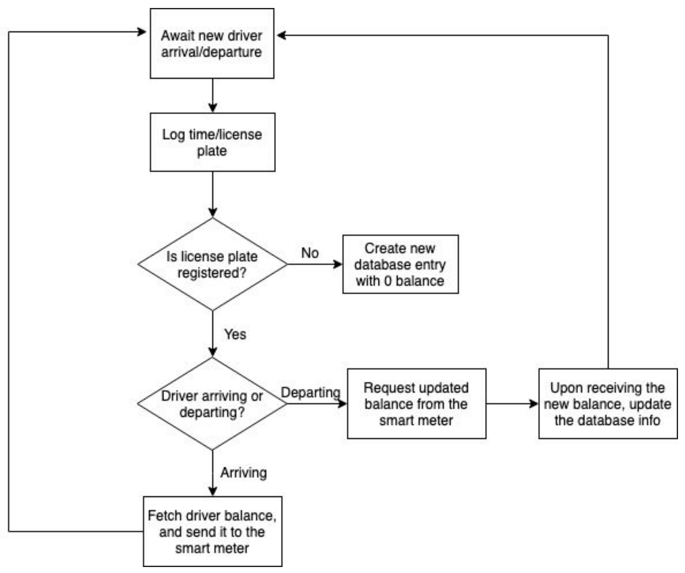

Await new driver arrival/departure
Log time/license plate
Is license plate registered?
No
Create new database entry with 0 balance
Yes
Driver arriving or departing?
Departing
Request updated balance from the smart meter
Upon receiving the new balance, update the database info
Arriving
Fetch driver balance, and send it to the smart meter

Figure 6: Flow of data on the server side.

2.4.3 Server and Database

Upon a driver arriving, a request is made to query the database for an existing plate. If an existing is balance is found, the balance is sent to the smart meter and converted into a number of minutes the user is allowed to park for. If the balance reaches zero, or a violation is committed at any point during the flow, an interrupt is generated and if the issue is not cleared within 15 minutes, University Parking is alerted of the incident, and the driver's account is updated with the corresponding incident message.

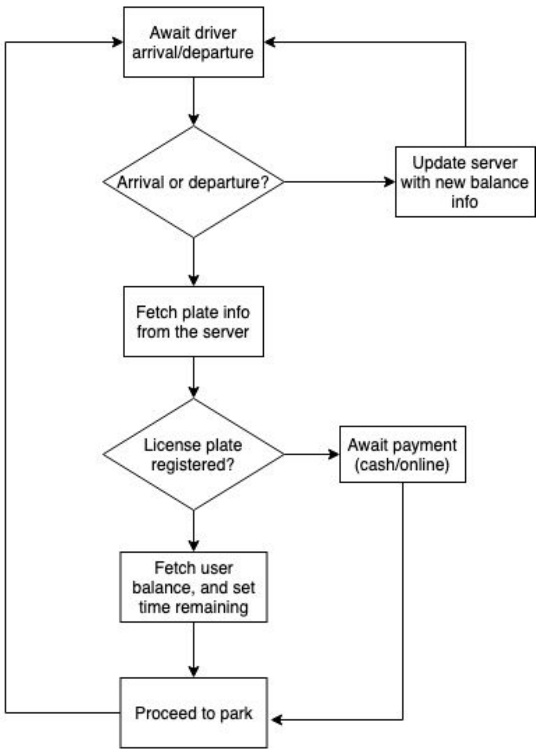

Await driver arrival/departure
Arrival or departure?
Update server with new balance info
Fetch plate info from the server
License plate registered?
Await payment (cash/online)
Fetch user balance, and set time remaining
Proceed to park

Figure 7: Software logic flow on the meter/client side.

<u>2.4.4 Plate Recognition Model</u>

The plate recognition model will consist of multiple models. The first model consists of detecting objects with the shape of a license plate, which means this model will be trained to recognize a rectangle of a specific size, edge, and of corner shape (since license plates have distinctly rounded corners). The second portion of the model consists of optical character recognition. Once a license plate-shaped object is detected, there needs to be a model to recognize text on the license plate itself, and associate this text with ASCII characters in order to perform lookups based on the license plate. We will integrate help from the OpenALPR library, which contains models for plate-shape recognition and OCR, which is built on top of OpenCV and written in C++.

2.5 Tolerance Analysis

<u>2.5.1 Raspberry power supply</u>

Since we intend to wake up Raspberry Pi while necessary, we have to build this module to automatically turn on the raspberry pi. (we could automatically turn it off in the program) By building this module, we could turn on the Raspberry pi depends on the instruction of MCU. The MOSFET would works perfectly as a digital switch, but it still has internal resistance. By roughly measuring the current through a working Pi, we tested and get the resistance of Pi would vary 5 ohms to 10 ohms. If we apply 5.1 V as VDD, and we want to ensure Raspberry could ensure 5 V voltage to work:

*5V = 5 ohms / (5 ohms + Rd) * 5.1V*

Then, Rds must be smaller than 0.1 ohms. We now find a MOSFET IRLB8721PbF, with smaller internal resistance.

2.5.2 Distance estimation

Accurate estimation of the distance between the car and the meter is important for first, assisting the driver to park car, and second, help to maintain car at a similar distance from the camera so that the plate number can be recognized with higher accuracy. We have not tested with real cars yet. From datasheet of ultrasonic sensor (HC-SR04), the measuring angle is 15 degree, and the measuring accuracy is 3mm. Ultrasonic sensor measures the distance by sending ultrasonic wave to the object and calculate the time until it receives the reflection of the sound from the object. We calculate the distance by following the equation:

$$distance = 344m/s * \frac{time}{2}$$

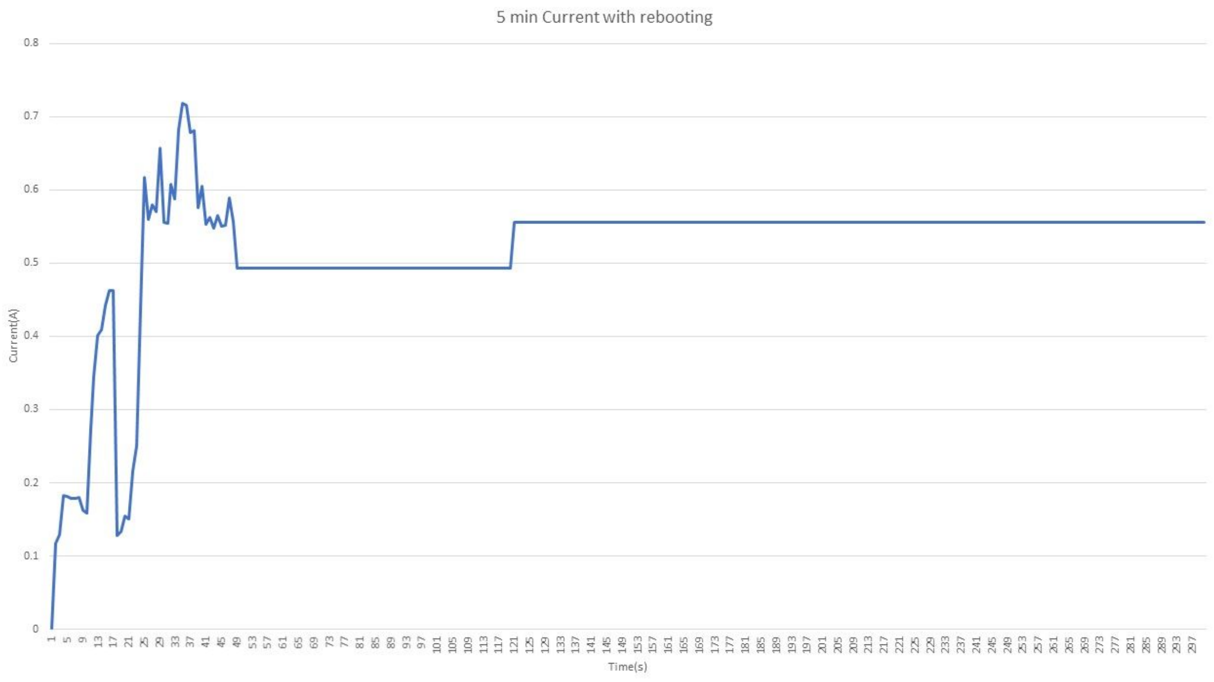
We use 344m/s for the speed of sound in the air.

Because of the different reflection surface and different height of cars, more than one sensor might be needed to be placed on different positions.

2.5.3 Plate recognition algorithm accuracy and response time

Our model needs to accurately recognize the plate number in a limited time frame. Due to our implementation, we will start the raspberry pi board first if MCU detects a car parking into our spot. The starting time of raspberry pi takes less than 2 minutes. The recognition algorithms takes less than 1 minute. In total of 3 minutes, we should have output before the driver is ready to leave. The accuracy of openALPR (4), as it is reported, has reached 99.2%. We believed that in our case, the actual accuracy will be higher because we limit the possible positions the plate number will show up in the camera.

2.5.4 Power consumption savings



Above is our recorded data for current while Raspberry booting, the voltage is fixed as 5.01 V. The process assume our system working 5 mins for each car coming. And the standby current for Raspberry Pi is 0.493A and assume the average working current is 0.550A . So, if we leave the Raspberry on all the time. In one busiest hour(4 cars come and leave), the energy consumed would be:

$$E = 20 * 60 * 5.01 * 0.493 + 40 * 60 * 5.01 * 0.550 = 2963.9 + 6613.2 = 9577.1 J$$

However, If we let Raspberry Pi work while necessary, we would consume less energy(average working current with rebooting is 0.517A):

$$E = 20 * 60 * 5.01 * 0.517 = 3108.2 J$$

So, if we cut the power to raspberry pi when we don’t need it for plate recognition. In total of one hour, we save *6468.9 J*. In practice, we can save more power since there will be fewer cars arriving at one parking spot for every hour.

3. Cost and Analysis

3.1 Cost Analysis

- Labor:
  Rate: $50/hr
  Bo Wang: 50 * 2.5 * (10 * 3) = $3750
  Christopher Santoso: 50 * 2.5 * (10 * 3) = $3750

Ximin Lin: 50 * 2.5 * (10 * 3) = $3750

- Parts:

Name (part number)	Quantity	Unit price($)	Subtotal($)
Raspberry pi board model 3 b+ from amazon (RASPBERRY PI 3 MODEL B+)	1	35	35
Camera Module v2 Raspberry Pi from amazon (913-2664)	1	23.9	23.9
SanDisk Ultra 16GB Ultra Micro SDHC UHS-I/Class 10 Card from amazon	2	7.2	14.4
CanaKit 5V 2.5A Raspberry Pi 3 B+ Power Supply/Adapter from amazon	1	9.99	9.99
LCD screen (EA DIP162-DN3LW)	1	17.5	17.5
PIC microcontroller (PIC16F877A)	1	4	4
Ultrasonic sensor hc-sr04 (3942)	1	3.95	3.95

-

3.2. Schedule

Date (by end of week)	Group Scheduled Progress	Ximin Lin	Chris Santoso	Bo Wang
2/18	Simulate ultrasonic sensor and part of power module, cable modification for raspberry pi. Develop plate number recognition model.			Simulate the power module, do the cable modification for raspberry pi.
2/25	Programming MCU for ultrasonic sensor and LCD, simulate on breadboard	Simulate Ultrasonic sensors, LCD display	Begin setup of the WiFi communications with the server.	Combining all parts on breadboard and debug
3/4	Design PCB on Eagle for the first wave PCB order.	Design PCB	Continue setting up WiFi-enabled communications.	PCB design and simulation
3/11	Finish plate recognition model and testing. Build the structure of software in Python. (PCB modification)	Research Plate recognition model and testing on raspberry pi	Set up the UART communications between the Raspberry Pi and the Microcontroller.	Testing the module, help with python code.
3/18	Spring break (no scheduled work for now)	Continue working on plate recognition and PCB modification	Continue working on the UART communications between the Raspberry Pi and the Microcontroller.	

3/25	Coding for data transfer between smart meters and server. Build the database and user interface of our service.	Help Chris build the server connection with database	Begin the back-end server connection with the database.	Finish building the database and test the connection.
4/1	Continue and design a custom cover for system. (may order a powerful battery to supply the project for outdoor testing)		Continue the back-end server connection with the database.	Design the cover for our meter, and find out the solution for testing battary.
4/8	Testing			
4/15	Testing			
4/22	Testing			
4/29	Testing			

4. Ethics and Safety

One of the primary concerns to be aware of is the fact that user data and their credit card account information is being stored and charged while utilizing our service. For this reason, it’s imperative that user information is properly encrypted and transmitted in a safe and secure manner. This will require us to research our options for when it comes to storing and transferring data.

An issue brought up during our discussions revealed that a similar automated parking collection meter was installed in Palisades Park (1). As a result, user complaints were numerous, and stores lost customers because they simply didn’t want to deal with the complications of these new meters. Furthermore, the city was able to raise a lot of money from drivers going over time, since tickets would be automatically billed to their address. This practice seems to clearly violate IEEE ethics code #5: “to improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies, including intelligent systems.” Because the use of their meters was confusing and unclear, many users suffered fines, which only seems to exacerbate the concern of automated processes. We seek to improve user parking experience with a more fully-featured automated driver parking assist.

Our meter will provide clear signals to the driver when a violation has occurred, and one of our biggest goals is to make the user interface as simple as possible. To reap benefits from users' confusion is unethical, so it's very important that our design choices make sense to the driver and that our signals and instructions are easy to follow.
IEEE ethics rule #6 (“to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations”) brings up another concern about storing and transmitting user data. Only one of us has experience with storing and retrieving encrypted user data for public services, so in doing so, we acknowledge that our knowledge in the field is limited, so in order to prevent unsafe practices, we plan to consult online resources and utilize existing tools for secure transfer and storage of user data.

5. Schematics

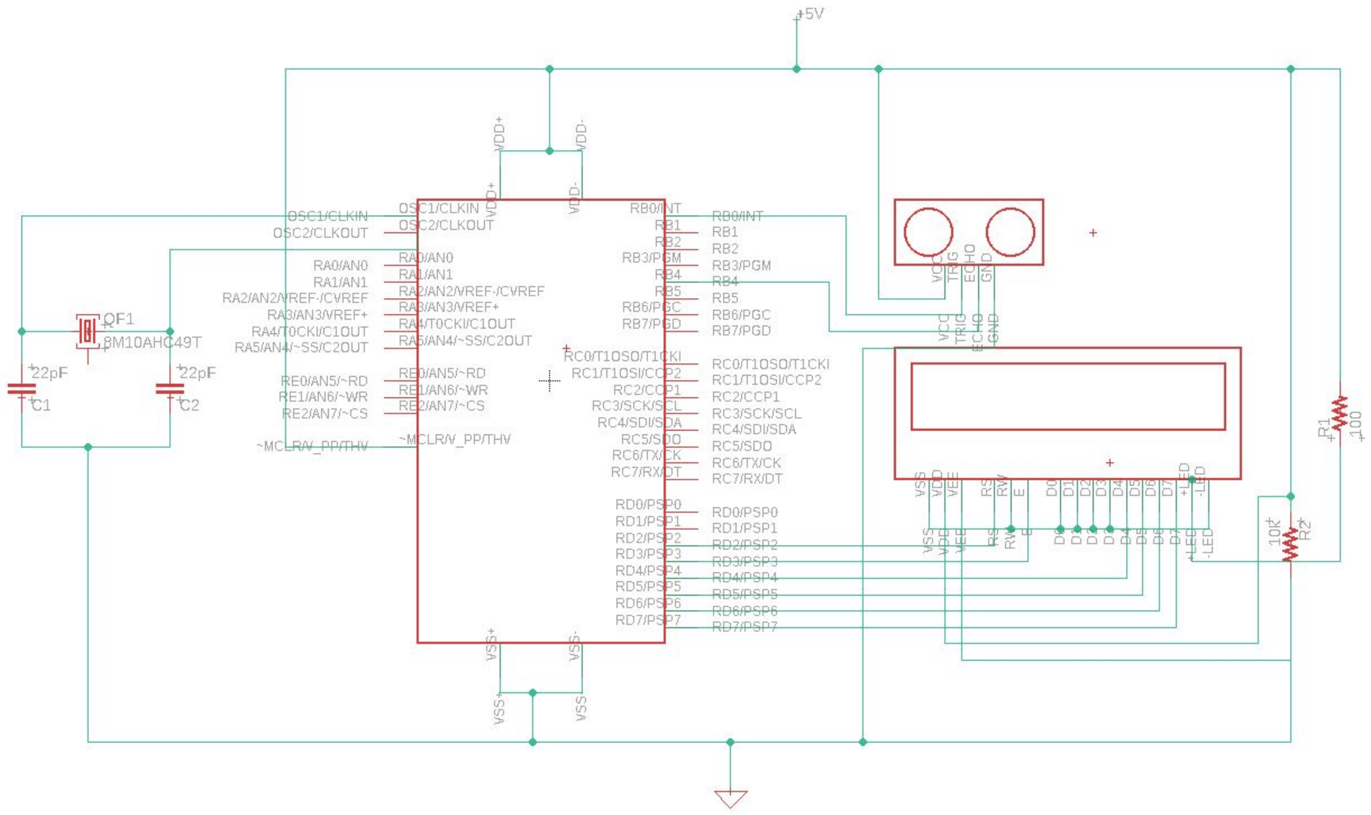
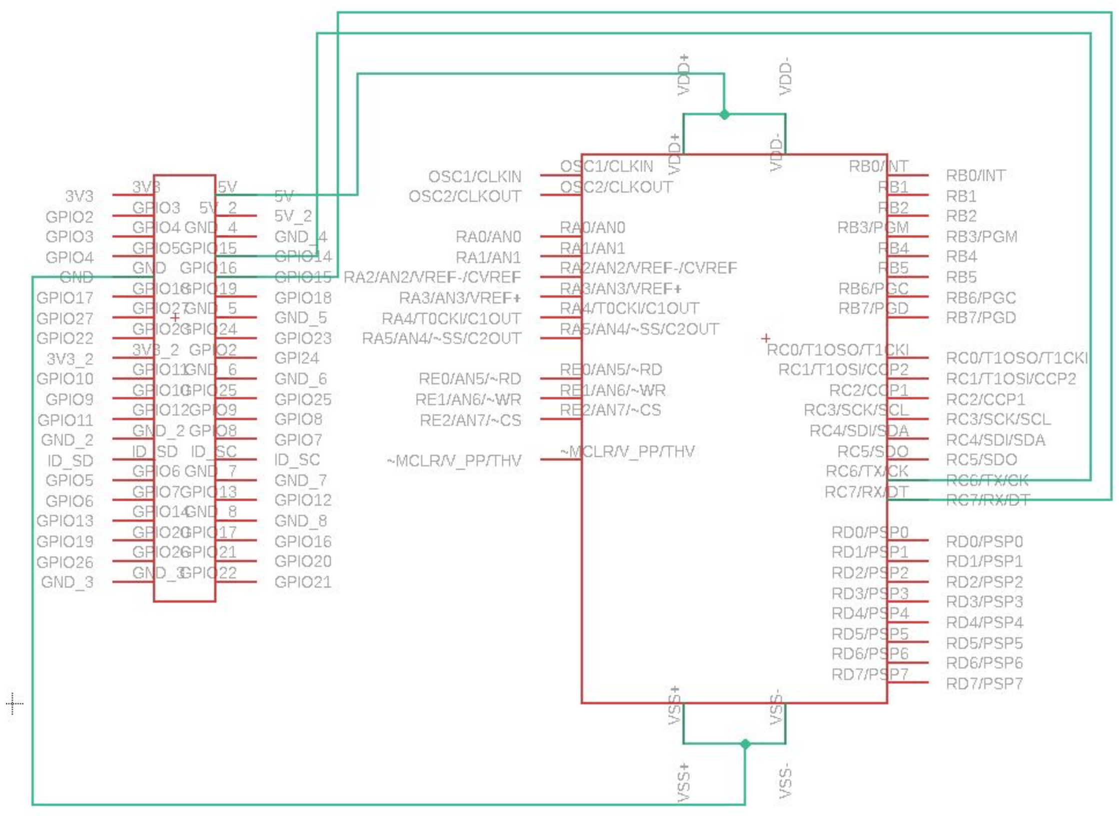


Figure 9. PIC16F877A microcontroller, LCD, and motion sensor connections circuit diagram.



<u>Figure 10.</u> Raspberry Pi B+ to PIC16F877A UART connections.

6. Citations

1. northjersey. (2019). Palisades Park: Digital parking meters a chance at a camera windfall. [online] Available at: https://www.northjersey.com/story/news/bergen/palisades-park/2018/06/19/palisades-park-digital-parking-meters-camera-windfall/710805002/ [Accessed 3 Feb. 2019].
2. Learn.adafruit.com. (2019). How PIRs Work | PIR Motion Sensor | Adafruit Learning System. [online] Available at: https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/how-pirs-work [Accessed 4 Feb. 2019].
3. MaxBotix Inc. (2019). Ultrasonic vs Infrared (IR) Sensors - Which is better? - MaxBotix Inc.. [online] Available at: https://www.maxbotix.com/articles/ultrasonic-or-infrared-sensors.htm [Accessed 4 Feb. 2019].
4. Officer.com (2018). OpenALPR Software Upgrade: Brings Accuracy Up To 99.02%. [online] Available at: https://www.officer.com/command-hq/technology/traffic/lpr-license-plate-recognition/press-release/21031077/openalpr-technology-inc-openalpr-software-upgrade-brings-accuracy-to-9902 [Accessed 18 Feb. 2019].