Team Name: Best Team

Demo Video: https://mediaspace.illinois.edu/media/t/1_c4zjk3js

Keyi Shen, Haina Lou, Shuting Shao

keyis2, hainal2, shao27

CS 437 Final Report

Smart Parking System

Part 1: Motivation

We have noticed a number of problems with the parking system in some communities. Firstly, vehicles entering the car park but cannot find a space easily. When the terrain in the car park is complex and the driver is not familiar with the car park, or when the car park is close to being full, it will be not easy for the driver to find an available parking space. Spending too much time on finding the parking space, which is an inefficient parking, not only is a waste of time for drivers, but also may lead to congestion in the community. Secondly, there are no measures to ensure the safety of vehicles after they have entered the car park. There is a possibility that someone from outside the community may enter the car park and flee the scene after intentionally or unintentionally breaking into a vehicle. Thirdly, some of the communities provide too many parking spaces for visitors to get more money for parking, which makes the residents lose the convenience of getting a parking space. Sometimes, residents may need to park far from their home.

To solve the first problem – inefficient parking, our solution is to provide a visualized available parking position for drivers. For residents who often park in the car park, they can use an app to

see the distribution of real-time available parking spaces. For visitors who don't use the parking app, they have another option. They will be guided to an assigned parking space by following the updating introduction and marking in the whole car park.

To solve the second problem – safety, our solution is to strictly limit the access to the car park. People who don't have the access permits will not be allowed to enter the car park. Residents of the community have access permits; visitors to the car park are granted temporary access through an automatic register that measures information when they enter the car park. Anyone who wants to enter the car park will need to register their basic information at the entrance. Besides, there are sensors assigned at every parking space, it will trigger the alarm if any anomalies are detected around the vehicle or if the vehicle is vandalized.

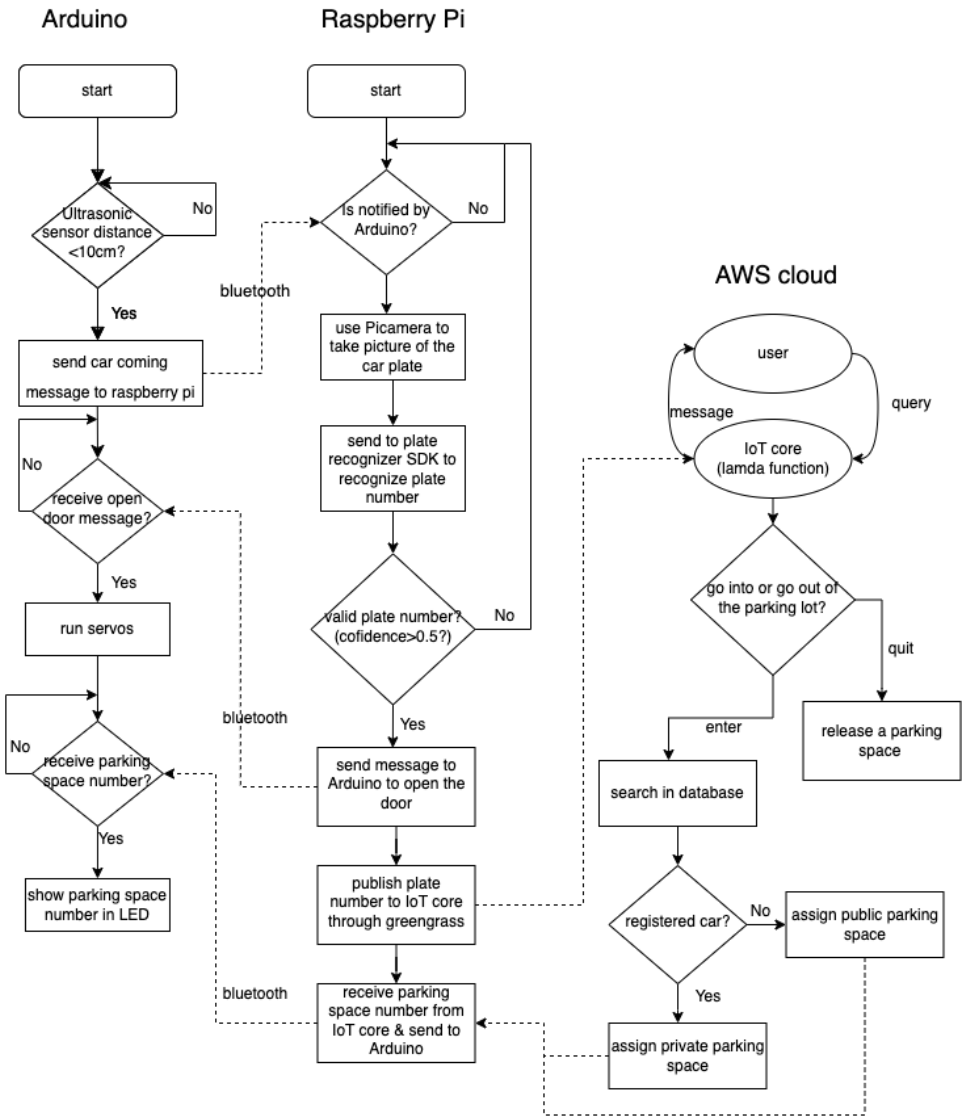
To solve the problem – vehicle dispatching, we're going to build a smart parking dispatching system. We will give the parking spaces to visitors on the condition that resistors are given priority. Resistors can reserve the parking spaces on app or set some spaces as their private parking spaces. The parking dispatching system will allocate the parking spaces separately for resistors and visitors.

This smart parking system can solve many problems of the current parking system. It improves the parking system effectively from three aspects – visual parking spaces, vehicle safety, smart parking dispatching.

Part 2: Technical Approach

The flowchart of the project is as followed. We have two types of devices which are shown with color purple. One is user, which will query parking lot information, its parking location and parking space number. The other one is controller, which mainly control the parking entrance system. When the ultrasonic sensor detects the car is coming, it will notify raspberry pi by

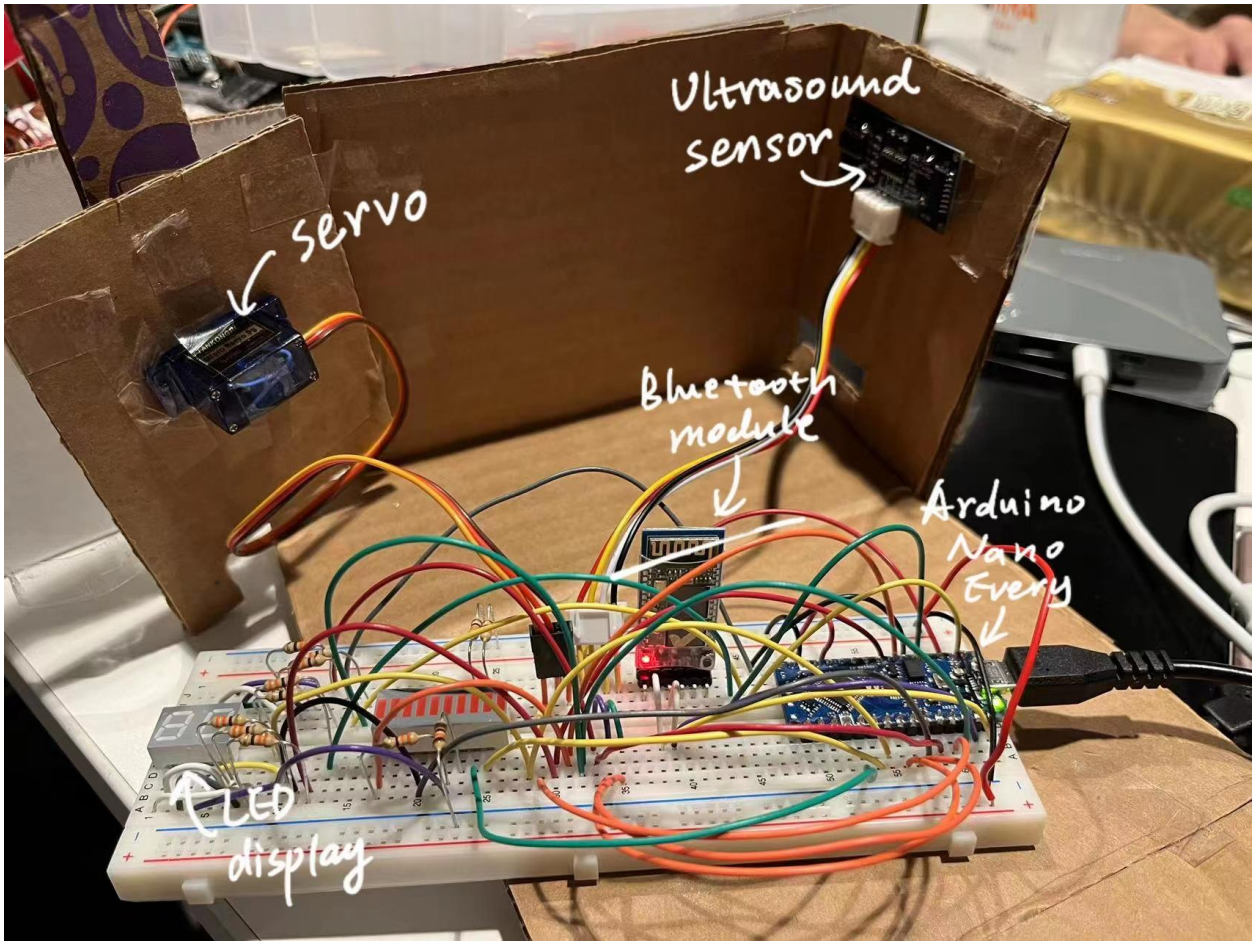
bluetooth and raspberry pi will notify arduino when the door of parking lot should open. The greengrass core is on the raspberry pi which carries the responsibility to communicate with IoT core. We can clearly see from the flowchart with color blue, the raspberry pi will send the plate number to lambda function and receive the parking space number message back.



Part 3: Implementation Details

Arduino Part

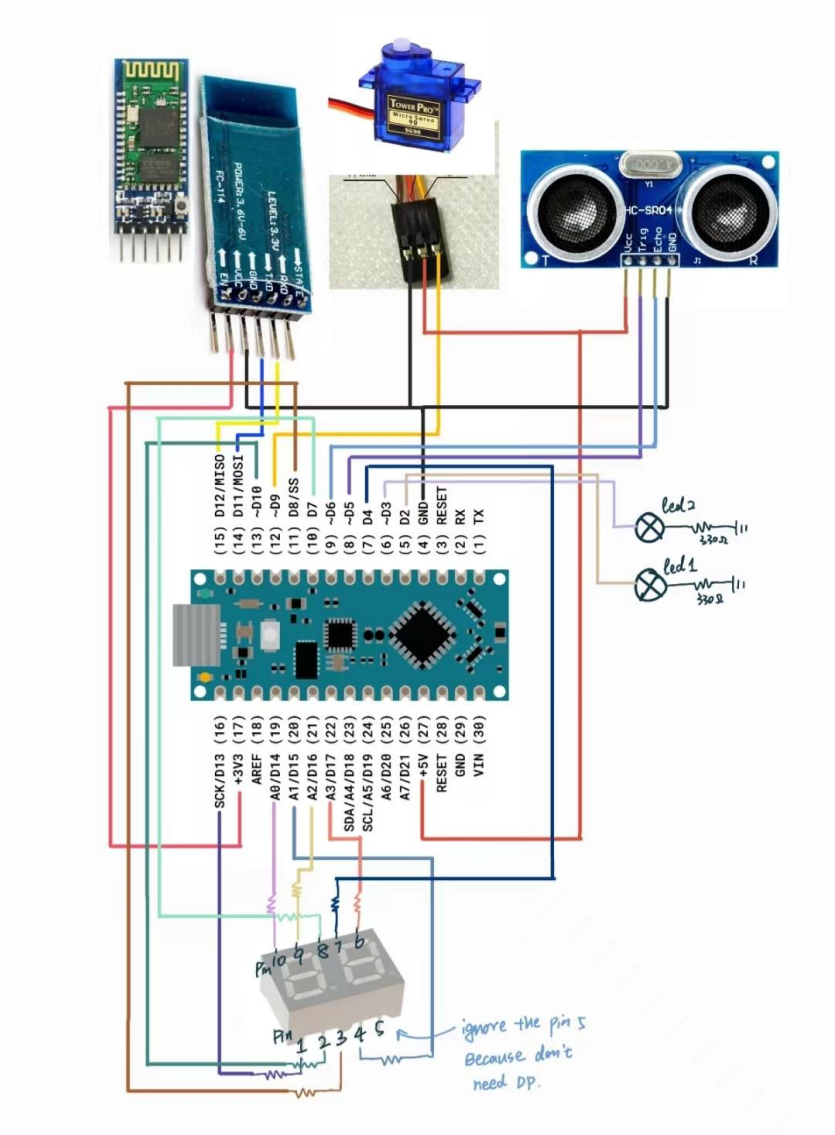
Because we want to build a small and compact circuit for the hardware part, and the function we want to implement for this part is not too complicated, we choose the Arduino Nano Every which is small and can be plugged into the breadboard to be our microcontroller for this part. In this part, we achieve the communication between arduino and Raspberry Pi through Bluetooth, detect the approaching object through Ultrasound sensor, control the motion of servo, and display the parking space number on a 2-digit 7-segment led display. Our circuit is as follows.



Parts used:

Arduino Nano Every (Atmega4809)
HC-05 Bluetooth module

LTD-2601G 2-digit 7-segment led display
SG90 Micro servo
HC-SR04 Ultrasonic sensor
2 x Leds
Jump wires
1 x Breadboard
11 x 330-ohm resistors



The ultrasonic sensor keeps on detecting the object in front of it (every 1000ms). If there is an object (like vehicle) approaching the sensor, the ultrasonic sensor detects that the distance is smaller than 10cm, the arduino will let the bluetooth module send a ‘1’ message to the receiver. The message is to tell the system that there is a vehicle waiting for the permission to enter the

parking space. The bluetooth module will always wait for the orders from raspberry pi which plays a role in GreenGrass of the system. We make a road gate to limit the entering of the vehicles. The raising and lowering of the road gate is controlled by the servo. If the arduino gets the ‘success’ message, which means the permission is given, it will make the road gate raise and let the vehicle pass. If the ultrasonic sensor detects that the vehicle has passed the entrance (it will need to make sure that there is no vehicle at the entrance to avoid the road gate hitting the vehicle), it will wait for 1000ms and then lower the road gate. If the arduino gets a message that is a number, the number represents the parking space number that the vehicle is assigned to. The parking space number will be shown on the led display to let the driver know where he should park his vehicle. The led display will show for a while and then turn off.

ALPR parking system with Plate Recognizer’s SDK

To develop a smart parking system designed for the community with high security, we use ALPR to detect and recognize the license plate number of the car at entrance and store it in our database by AWS iot platform. Because our time is limited for training the recognition model by ourselves, We search online and find one blog writing about using a service from plate recognizer(Codrin's Ideas) and use this method. We use the service from plate recognizer, which can help us to get the result of recognizing the license plate number and the time. At the start, we open the camera, open the ALPR service, connect with the arduino by bluetooth serial port and wait for the arduino to send a message that a car is approaching. When receiving a “1” which means an ultrasonic sensor has detected something very close and then uses the pi camera to take a picture, it will write a “1” in state.txt. At the same time, we use another terminal to run a script which will monitor file

changes. When the file rewrites, it will send the picture captured by the pi camera to the plate recognizer platform and store the detection result back in the file with json format. At last, we process the data, extract the useful data like plate number and timestamp, and send it to AWS IoT core by greengrass core. Then we subscribe to the cloud topic and receive the message whether we should open the parking lot door or not, and send the assigned parking space number message to Arduino to make it show on the LED. The code is attached in the end.

IoT core and lambda function

We use AWS IoT to build our cloud platform. AWS IoT also helps us emulate and verify our system in lack of hardware. Otherwise, we need more physical devices acting as users' devices.

In our implementation, we create different types of devices in the Greengrass group. One type of device is the users' apps which are used by users and can communicate with the cloud, another one is the local controller which includes a camera and electric gate and is specified before. Instead of using an EC2 instance, we use the Raspberry Pi as the Greengrass Core, which can build the communication between the devices and cloud. Greengrass is used since the devices do not have a connection with other devices directly in reality. On the cloud side, we have lambda functions and datasets as scheduling and database units.

The first one is the database unit which stores all basic information of our system, including information of users, vehicles and carports and the running log. We maintain 2 datasets, one is to

store information of users, vehicles, and carports. So, we can query. Another one is to store the running log.

Our system is a half-opening system which provides parking service for both registered and temporary vehicles. We assume that vehicles have two types: registered and temporary(guest). For a registered vehicle, we have the personal information of the vehicle owner, and it has a pre-assigned carport. i.e., the owner owns a private carport to park his vehicle. Also, we do not limit the number of vehicles and carports owned by one user. So, we cannot assume that there is a one-to-one mapping relationship between users and vehicles, carports.

To support the feature mentioned above, carports in our system can be divided into private carports and public carports. A private carport is owned by someone and can be assigned to the particular vehicle, any other vehicles can never park into it. Instead, the public carports are not owned by anyone, so any vehicles can park into it.

With the first dataset, we can find the relationship between vehicles, carports, and users and the current state of a particular vehicle (the index of carport occupying) by query. This dataset is the base of our scheduling – we need to query and update it when there are some vehicles entering or quitting. In our implementation, to keep simplicity, we maintain a local version inside the lambda function, which acts as the scheduling unit and will be specified below. This implementation is somewhat reasonable since compared to the running log, the data size of the first dataset is relatively constant, i.e., after initialization, we will not add a great deal of users or vehicles. Besides, the access and modification of these data is frequent. So, it also helps to keep the performance. This vocal dataset is implemented by the Pandas.DataFrame structure, which is a

great applied structure in practice and conveniently to organize data as dataset and deal with big data.

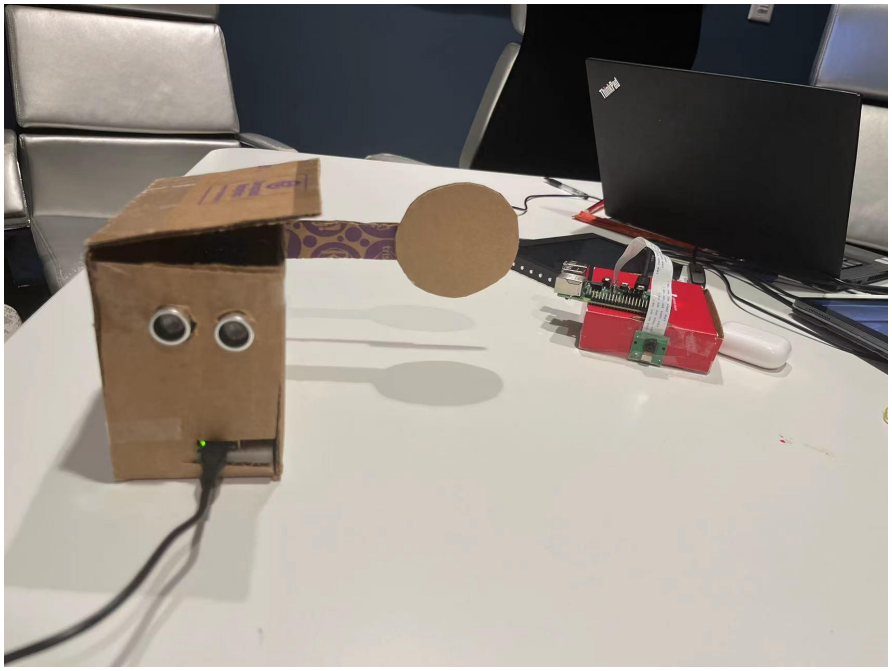
The second dataset stores the running log. The features of this dataset are that 1. The size is monotonously increasing and after the data is generated, the access and modification of it is infrequent. So, we don't have a local version in the lambda function, or alternatively, we can hold a summary or a size-limited dataset and clear it when it is full. From a practical point of view, this running log is necessary when security incidents happen and also helps to do some data analysis.

The scheduling unit, which is the core of the cloud platform, is implemented by the lambda function. The lambda function subscribes to at least two types of topics. One type of topic is related to the users' devices such as the apps. Another one is related to the local (as opposed to the cloud) controller. The basic data from the local controller involves the license plate number (vehicle id) and timestamp. We need to deal with the basic conditions like there is a vehicle entering or quitting. When there is a vehicle entering, the scheduling unit needs to judge the type of the vehicle (registered or temporary) to assign a private or public carport. For registered vehicles, we try to assign the corresponding carports stored in the dataset first. For guest vehicles, we assign an available public carport. To avoid the possibility of traversing all the public carports to determine the available carport, we maintain a "next free array" and "free id" to accelerate. Its time complexity is in O (1) rather than O (n). when a vehicle quits, we also need to unassign the occupied carport. The result of the operations will be sent back to the local controller and also sent to the running log. About the topic with users' app, we support two basic

features: users can query the situation of carports to find whether there are any available carports, and the cloud may send some confirming messages for safety, for example, another person drives a vehicle with the same vehicle id as one user. If we get the approval from the user, we can assume there is a user’s acquaintance driving the user’s vehicle and can allow the vehicle to enter. Otherwise, we reject the entry request for security.

Part 4: Results

We have basically implemented the smart parking system. After the car enters the entry, it will be detected and the camera will take a picture of the front of the car and upload the photo to identify its license plate. The car's information is then recorded and the road gate is raised to allow the car to drive in. Once inside, the car will see the parking space it has been allocated on the led display.



For the ALPR part, the point which still needs improvement is that it is hard to control the frequency of taking pictures. Because our camera instruction is triggered by the ultrasonic sensor and the ultrasonic sensor is not always so accurate. Therefore, there are always fake alarms if we implement this device in the real world and every time the pi camera captures one picture, it will send it to the Plate Recognizer platform. It is inefficient because a lot of pictures may not include the true car. To improve this, we need other sensors' help like a pressure sensor to send the message to take the picture and improve the logic of taking pictures.

There is a tricky problem which puzzled us for several hours when we deploy our cloud platform and do the simulation. As mentioned before, to implement the local version of the dataset, we use the DataFrame, which is included in the Pandas library. But when we run the lambda function, the running log of the lambda function shows that "no module named pandas" which means the Pandas library is not installed in the running python environment. Since we are using the lambda function, we first search for how to install Pandas into the AWS lambda function. After some attempts, for example, install Pandas and compress it with our lambda function, add a

layer including the virtual python environment with Pandas, we can use the pandas in the test console online, which means the solution works well. However, after deployment we still get the same error. Finally, we find that since we are using Greengrass, the Raspberry Pi acts as the Greengrass Core. Then it will run the lambda function locally. So, what we really need to do is installing Pandas on Raspberry Pi rather than deploying online.

After solving the problem, we want to say that we almost reached our goal for this project. For the cloud platform, it can deal with all data from the local controller and users, give the appropriate response based on some scheduling strategies and also store all data on the cloud for archive and analysis.

Part 5: Skill-building and innovation:

Keyi: I implement the cloud platform. Even though we use the AWS IoT platform as what we do in Lab4, I still learn new things. except designing the scheduling strategy to fit all requirements, I need to maintain the datasets we need. In this process, I get more familiar with Pandas, and to get better performance, I try to do some optimization in querying the dataset. During the deployment on AWS, I learn more about the framework of IoT and Greengrass, how to reasonably configure lambda function according to the demand and how to configure the running environment on lambda function such as with the AWS Cloud 9 and lambda layer, even though it doesn't solve the problem above.

Haina: I implement the Plate Recognition and controller part. My work mainly relies on raspberry pi and it is like an information transfer station which builds the communication between devices and the cloud platform. To accomplish plate recognition, I search the internet

and get in touch with the CV object detection model which is also quite often used in industry. From this experience, I have learned how to use the API of SDK and this service is quite useful which I am willing to use in my future in the other project. Also, I send and receive data by bluetooth connection and greengrass core of cloud service. I think they are very useful tools and I get more familiar with bluetooth API and also the overall concept and structure.

Shuting: I implement the Arduino part. In this part, I learned how to use the arduino, how to build the communication between arduino and raspberry pi by using a bluetooth module, how to control a servo to a certain position by setting the angle, and how to display numbers on the led display. I design the circuit to implement those different functions in order. Also, I made a small box to hold the circuit, so that the servo and the ultrasonic sensor can be assigned at the right position, and the circuit can be protected well. This is a really interesting process to build this hardware part, from designing the circuit to debugging the code to achieve the function and effects we want. I have learned many new things in this process. The study of arduino in lab5 is very helpful for this part of the final project. Finally, I think arduino nano is a really useful tool to implement some interesting but not too complicated functions, it can deal with uncomplicated data effectively, but it is easy to learn how to use arduino.

In conclusion, we successfully build the system from the low-level hardware to high-level software, from the local devices to the cloud. We get more experience in circuit & hardware assembly, Arduino, CV object detection, communication building between different devices, cloud platform building and data processing… Hence, we extend our skill set well based on prior knowledge and what learnt in this course.

Finally, we would like to express our gratitude to Professor Caesar and all teaching assistants for your hard work and patient guidance during this semester.

Citation

Datasheet of the LTD-2601G led display.
https://optoelectronics.liteon.com/upload/download/DS-30-95-286/D2601G.pdf
Datasheet of Nano Every.
https://docs.arduino.cc/static/5ae24b68bef5c4d925b6b8d24b0be06d/ABX00028-datasheet.pdf
HC-05 Serial Bluetooth Module. https://electrosome.com/hc-05-serial-bluetooth-module/
Get Started with HC-05 Bluetooth Module & Arduino.
https://create.arduino.cc/projecthub/electropeak/getting-started-with-hc-05-bluetooth-module-arduino-e0ca81
2-Digit 7 Segment Counter(00-99) with Arduino in Proteus. May 18, 2020.
https://microdigisoft.com/2-digit-7-segment-counter00-99with-arduino-in-proteus/
Codrin's Ideas. DIY Raspberry Pi ALPR Parking System with PlateRecognizer's SDK. Jun 30,2021.
https://codrinsideas.medium.com/diy-raspberry-pi-parking-system-with-platerecognizers-alpr-8e2254298917