Final Report: IoT Home Security Network

Lorenzo Bujalil Silva, Logan Cudia, Joshua Cudia

ECE 479: IoT and Cognitive Computing

1 Track

For the final project, we decided to follow the IoT track and build a home security system. We attempted to incorporate many different platforms into our design, which include Raspberry Pi, NVIDIA Jetson Nano, and Amazon Web Services. Throughout the building process of the project we met many difficulties and it forced us to modify our design various times. Despite encountering these problems, we we able to successfully build a IoT based home security system.

2 Problem

Modern home security systems pose a significant security issue which may be exploited. Through our research, we found that most security systems fail to incorporate edge computing into their systems. Video and audio data in these systems are sent to the cloud in order to be processed, but this design causes issues with latency. Edge computing allows for object detection and facial recognition models to be run immediately and then a simple message may be sent to the cloud. There are some case where modern security solutions to have edge computing capabilities, but range in costs upwards of $1000. We set out to build an cost effective solution, which also provides powerful performance.

3 Solution

As mentioned above we designed a solution based on communicating information between powerful IoT devices and a database system. We initially had planned to build a system which has three Raspberry Pi's as edge nodes running detection/recognition models, one Nvidia Jetson Nano processing and delivering information between the database and the edge devices, and a cloud database system which can also deliver information to a mobile device. This system provides powerful, fast solution in order to collect information, run detection, and communicate important messages. We were able to successfully emulate a sophisticated modern home security system valued at only $150. We found that most modern security systems, are overvalued and can be replicated with some engineering.

4 Design

We covered the overview of the design in the previous paragraphs, but now we will go into a deeper analysis of the design. When we proposed this project we had our minds set on building a system with three edge nodes, one sink node, connection to Google Cloud Platform, and communication to a mobile application. When we started to build this system we encountered many issues. One of the first issues we encountered was with our edge Pi which was primarily handling the external sensors. With multiple sensors connected to this Pi, we soon found out that we needed an external power supply because the Pi's USB ports may not be able to provide enough power to run all the sensors and the Pi itself, which could result in unstable performance or failure. Our power supply consisted of the LM317 (adjustable linear voltage regulator adjusted to 3.3V) and the 7805ACT (a 5V linear voltage regulator). This circuit was powered by a 12V 24W wall wort, utilizing off-line power for less power loss and more longevity. The sensors connected to this included a 5v relay, 3.3V RFID with 3 key cards, and a solenoid electric lock. With the external power supply design, we can further scale this design to include more advanced, higher power sensors.

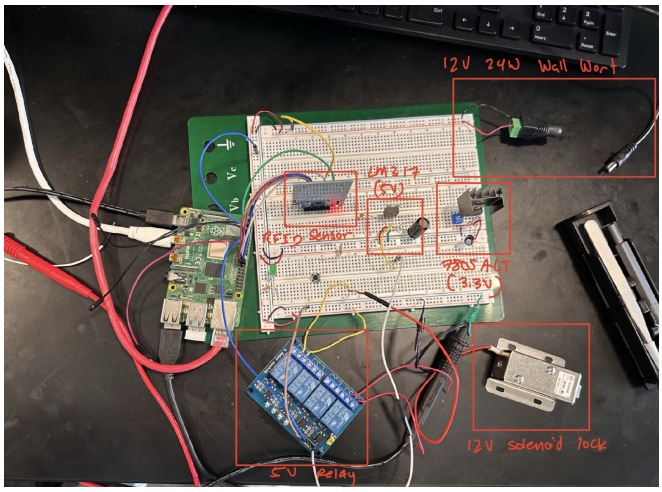

Figure 1: Circuit for Locking System

After the various issues we encountered with the NRF modules we decided to do more research on the communication protocols between embedded systems. We found that MQTT can easily be used to in order to communicate between the two IoT devices. We used a publish/subscription system through an broker to communicate between the devices and send messages to devices and the cloud.

At this point we were able to communicate between edge devices, and we needed to establish a communication to the cloud services. We proposed in the initial design to use Google Cloud Platform to host a database containing information for facial feature tensors. We were able to establish a Cloud SQL instance, but when we attempted to create a MySQL client on our sink node, we found some compatibility issues. Our sink node was unable to establish a communication with the cloud database, which lead us to attempting to use AWS IoT Core to establish a communication. We were able to communicate to the AWS IoT core by sending messages to the cloud, but we were not able to establish a communication back to the device. After many hours of attempting to set up communication to the cloud database, we decided to change the design to use a local database at the sink

node. With the database at the sink node we are able to communicate messages to the sink node using MQTT, and when a message is received at the sink node it will be able to cross reference the local database and depending if the similarity between feature tensors is small then the sink node will publish a message to edge device which is subscribing to the same broker. This message will communicate to the relay to open the door.

We have now created a communication between edge devices and to the local database which will decide to let the person in or not. Up to this point we were using the GoogLeNet convolutional neural network to create our facial feature tensors to be stored in the database, and we were using the EfficientNet convolutional neural network architecture to run object detection. We found that using these models are providing a baseline for our application, but we believed that we can do more. We then found that we could accelerate our object detection model using the Coral Edge TPU and we noticed a 62% decrease in inference times. We then sought out to better our facial recognition model, by parallelizing convolutional layers of GoogLeNet model using a NVIDIA Jetson Nano. At this point we met another issue with the Jetson. The Jetson JetPack SDK comes with Python 3.6.9 installed and our applications required to use a newer Python version. We spent hours attempting to remove the native Python version, but each time that we would update the Python version dependencies in the system would fail to work properly. During our lab demom, we explained our problems with the Jetson, and our TA informed us that we were able to use a Docker image in order to access the newer versions of Python without affecting the native version. If we knew this information before, we would have been able to accelerate the performance of our model thoroughly and the security system would provide more safety. However, we needed to continue to use the non-accelerated GoogLeNet model.

The final part of our design is too communicate the information if a person was detected or let into the house. We decided to use Twilio to send a text message to one of our phones notifying us if there was someone in the house or at the door. The twillio message will be sent when publishing a message over MQTT.

5 Evaluation

Our design offers faster, more efficient responses to real time situations. As previously mentioned, many edge based home security systems do not offer cost effective solutions. Our primary edge Pi's contains under $50 in sensors but have several layers which prevent intrusion and unwanted visitors. Furthermore, with our object detecting indoor Pi, if an intruder were to somehow penetrate through these layers of defense, the homeowner will be notified of peculiar movement within the house and be notified within seconds via text message. We are also using an efficient method to communicate between nodes to provide information quickly over WIFI. This information will quickly notify us when a person was detected or a person was at the door and will be let in. Our system beats other systems in the field of security systems because our system is able to provide a very quick inference times and a rapid delivery of messages.

6 Moving Forward

Now that we have identified the current challenges and limitations in our system, let's explore some improvements we can implement moving forward. One obvious improvement we can add is towards the hardware aspect of this design. At the "front door" node, we can implement more user-specific classifications. For example, the fingerprint sensor is a good line of defense for security systems due to their high level of accuracy and uniqueness of each individual's fingerprint. Unlike passwords or PINs, fingerprints cannot be forgotten or guessed, and they cannot be stolen or replicated as easily as other forms of authentication. Moreover, fingerprints are unique to each individual, making it extremely difficult for unauthorized users to bypass the security measures. Fingerprint sensors also provide an added layer of convenience, as users do not need to remember and type in passwords or PINs every time they want to access a secure system or device. There are many different types of sensors that can be added to an indoor home automation design to improve its functionality and efficiency. For example, temperature sensors can be used to monitor and control the temperature in different rooms of the house, ensuring that the temperature is set at a comfortable level while also reducing energy consumption. Similarly, motion sensors can be used to detect when someone enters or leaves a room, allowing for automatic control of lighting, heating, and cooling systems. Humidity sensors can be used to monitor the humidity levels in different rooms, which can help prevent the growth of mold and mildew. Smoke and carbon monoxide sensors can provide an early warning in the event of a fire or gas leak, helping to keep occupants safe. Finally, light sensors can be used to automatically adjust the lighting levels in different rooms based on the natural light available, which can help save energy and reduce electricity costs. Overall, adding sensors to an indoor home automation design can provide a range of benefits, from improved comfort and convenience to enhanced safety and energy efficiency.

Also regarding the systems detection models have room for improvement. With more time we would be able to accelerate our models with GPUs. We would also like to spend some time building a new model that runs sentiment analysis on the images determining if a person detected in the image is doing something suspicious.

Finally, we would love to provide more security to our IoT system building the security of the messages sent using the MQTT protocol. With more time, we would like to encrypt message using a public and private key system.

Overall, we find that there are many improvements that this system can receive and we hope to build on them after the end of the course.

7 References

A. Saxena, M. Tyagi and P. Singh, "Digital Outing System Using RFID And Raspberry Pi With MQTT Protocol," 2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-

SIU), Bhimtal, India, 2018, pp. 1-4, doi: 10.1109/IoT-SIU.2018.8519923.

Shifang Hai, Pengpeng Yu, Shuhua Zhou, Jianhua Zhao, and Jinshi Yang. 2022. Design and Simulation of Adjustable DC Regulated Power Supply Based on LM317. In Proceedings of the 4th International Conference on Information Technologies and Electrical Engineering (ICITEE '21). Association for Computing Machinery, New York, NY, USA, Article 33, 1–4. https://doi.org/10.1145/3513142.3513176

V. Patchava, H. B. Kandala and P. R. Babu, "A Smart Home Automation technique with Raspberry Pi using IoT," 2015 International Conference on Smart Sensors and Systems (IC-SSS), Bangalore, India, 2015, pp. 1-4, doi: 10.1109/SMARTSENS.2015.7873584

TensorFlow Object Detection Premade Model - EfficientNet
https://www.tensorflow.org/lite/examples/object_detection/overview
https://github.com/tensorflow/examples/tree/master/lite/examples/object_detection/raspberry_pi