Thermonitor

Author: Iris Wang, Jiamin Wang, Minji Kim: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—**Thermonitor is a smart, contactless thermometer that can be installed at gateways to different locations across buildings for large organizations, such as educational institutions and corporations, to remotely monitor their community members' temperature and facilitate contact tracing. The display of the Thermonitor will alert the users to scan their ID, properly wear a mask, and measure their temperatures once they are correctly identified as a member of the organization. Then the Thermonitor will collect the temperature data accompanied with one's profile detected by their RFID tag. This data will be uploaded to a web application, which will match the temperatures to individual profiles and log the records to allow organization admins to access and manage the logs.

***Index Terms*—Azure, Computer Vision, Face Detection, Internet of Things, Machine Learning, Mask Detection, NFC, RFID, STM 32, Thermometer, Web Application**

I. Introduction

Currently, COVID-19, a novel acute respiratory illness is plaguing countries all over the world. Infections are rising, with no view of a vaccine in sight. One of the earliest warning signs of infection is fever. It is critical to regularly monitor body temperature in order to protect others, especially for diseases like COVID-19 where one is contagious several days before showing any symptoms at all. By noticing changes in one's own body temperature, we can take immediate measures to prevent further spread of the virus.

Thermometers in the current market are either standalone kiosks or handheld. Handheld thermometers require another person to hold the thermometer from a distance of up to 4 inches, which fails to conform to the mandated 6 ft for proper social distancing. Standalone kiosks are currently over $2000, which can be unaffordable to be installed at every gate way. Currently, there is no viable product that is both safe and affordable; Thermonitor aims to be both. Thermonitor is an end-to-end device where the user presents a valid RFID tag that triggers the start of temperature measurement. A monitor is used to broadcast the video stream back to the user with a bounding box that surrounds the face. Thermonitor must measure the temperature within a distance of 50 cm (19.6 inches/1.64 feet) and this temperature is sampled multiple times within 3 seconds. An 85% facial detection and 95% mask detection accuracy is obtained. These algorithms ensure that the user is properly wearing a mask and alert them otherwise.

II. Design Requirements

The first requirement of Thermonitor is a capability to correctly read the RFID tag presented. This is necessary for us to send information from end-to-end on our device. It also serves as an indicator to know when the device should start scanning for faces and measuring temperatures. Secondly, it should be able to accurately measure the temperature of an individual standing in front of the device. It must also be able to properly detect faces in front of it using a bounding box to check whether or not the person is wearing a mask. All of this information must then be sent to our IoT platform for gathering and monitoring of data.

To verify that our design has met these specifications, we are performing several benchmark tests. To test the RFID scanner, test benches were created to make sure the device scans the IDs accurately and keeps track of unidentified individuals. We purchased sample RFID tags for testing the scanning functionality. We are aiming for 99% accuracy since misreads could occur occasionally.

To test the facial detection algorithm, we are putting both people and objects in front of the camera and see if their eyes or faces are detected by checking if a bounding box appears. In addition, we are testing faces with and without face masks in order to ensure it can still recognize people with only eyes as the tracking feature. If there are multiple people in the frame, the algorithm should only detect the main person in the front of the camera, the one who actually scanned their RFID. We are aiming for a 85% accuracy rate since the facial detection algorithm normally has a 95% accuracy, but since we are detecting it in real time video stream and we can only use a limited number of classifiers, we are lowering it. We are aiming for 95% mask detection rate since this algorithm is only being called after we determine there is a face. We are not aiming for 100% since we have a limited training set for mask detection.

To test temperature sensing, we are using various objects with different temperature ranges to test our degree measurement. We are aiming for ±0.2° Celsius from the intended temperature. Since sensor data needs to be transmitted, calculated, and displayed on the monitor, we are aiming for a 3 seconds time measurement. Handheld thermometers usually take around 1.5 seconds to display results and taking into account the processing time on the Jetson, we decided that 3 seconds is a good benchmark.

On the cloud side, we are making logs accessible to the admin and ensure that the correct profile is mapped to the RFID. We are using the sample RFIDs to test how easily accessible our external platform will be. To fully test our web application, we are conducting user testing on friends and family and adjust based on their feedback. We are using MQTT and TCP protocols that already ensure reliable messaging. However, we still should account for unexpected data transmission issues, thus we cannot guarantee 100% total transmission rate.

III. Architecture and/or Principle of Operation

We are using a microcontroller unit (MCU) to handle the logic behind our tag id verification. More specifically, we are using a STM32 Nucleo-L476RG board with a compatible expansion board X-NUCLEO-NFC02A1. The near field communication (NFC) expansion board is directly connected on top of the Nucleo board and the boards are powered through the micro-usb port. If a RFID tag is read, then a signal is sent from the Nucleo to the Jetson Nano to initialize temperature scanning and facial detection. Thus, the Nucleo is wired directly to the Jetson to turn off the auto-power on function provided by the Jetson.

An IR sensor is used to scan multiple accounts of temperature across a three second time frame. The Raspberry Pi camera module's video stream is fed directly to the Jetson to begin facial detection and then mask detection. The user must stand within a bounding box for accurate detection. All of this information is displayed on a monitor, as seen in Figure 1.

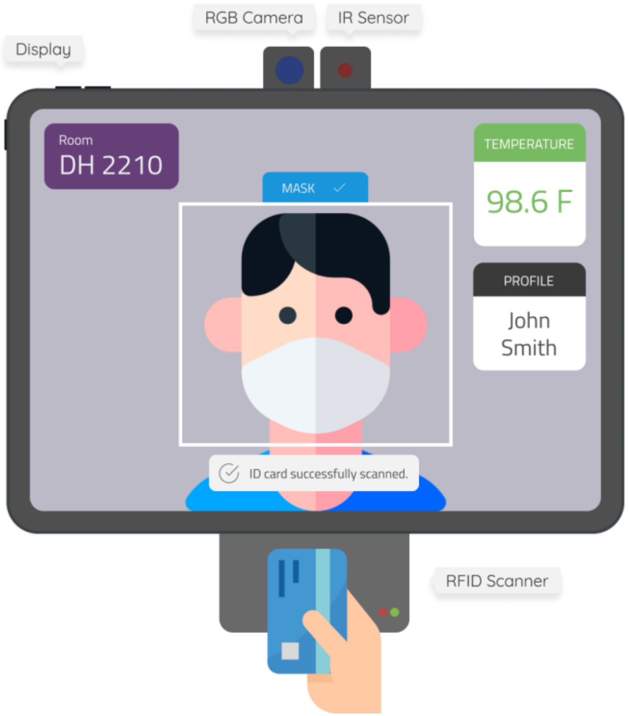
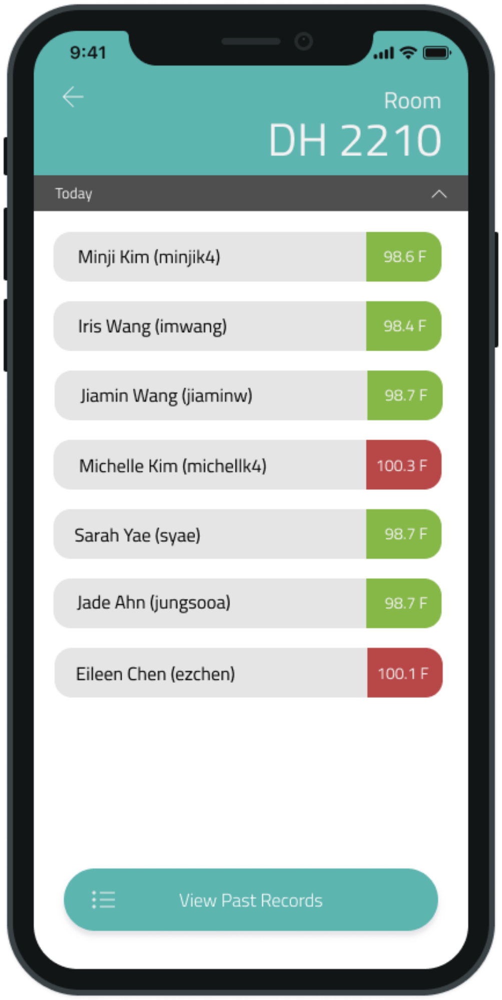


Fig. 1. Monitor display setup from the user perspective.

The Jetson Nano uses MQTT protocols to transmit user profile data across the cloud. This data is received by the Azure Hub, which is the cloud gateway for our IoT system. After unpacking the transmitted data and formatting it in the desired form, the data is sent to the back-end of our web application. This web application is hosted on Azure App Service, where they provide APIs to facilitate data transmission between Azure Hub and custom applications.

The web application is accessible through either a phone or a computer. The room name where the user scanned their ID, along with all user activity and corresponding temperatures are available to the admin. More information such as the timestamp becomes available once you click the individual record. Figure 2 provides an example of our web application wireframe.



Room
DH 2210

Today

Minji Kim (minjik4)	98.6 F
Iris Wang (imwang)	98.4 F
Jiamin Wang (jiaminw)	98.7 F
Michelle Kim (michellk4)	100.3 F
Sarah Yae (syae)	98.7 F
Jade Ahn (jungsooa)	98.7 F
Eileen Chen (ezchen)	100.1 F

Fig. 2. Web application wireframe with individual records.

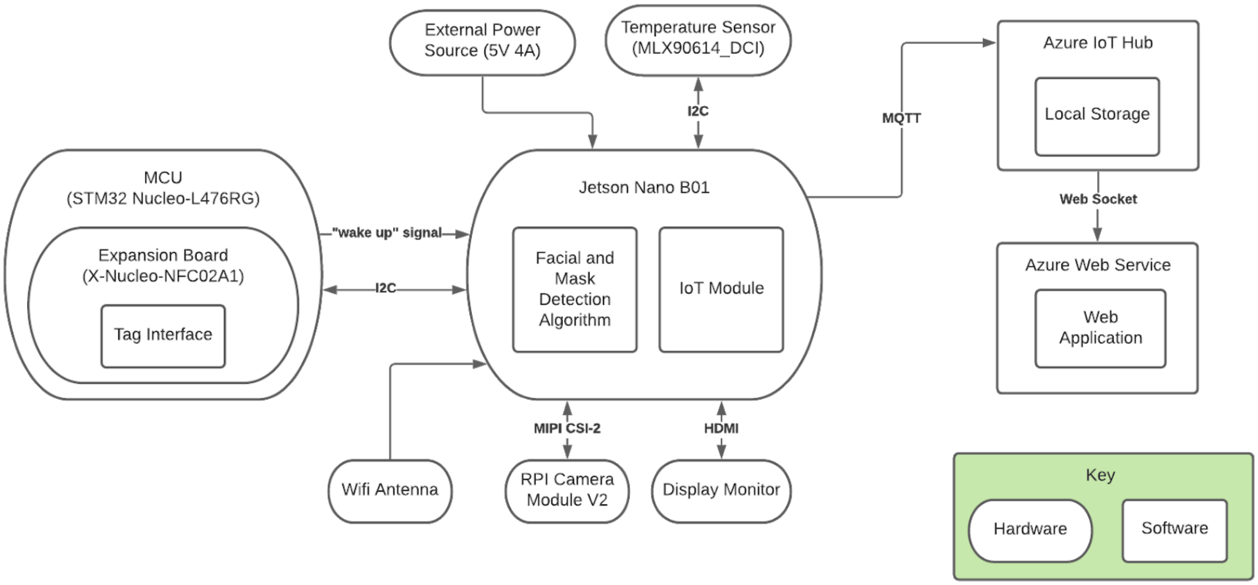


Fig. 3. Overall block diagram of our system.

IV. Design Trade Studies

These are some components we took into consideration when choosing which best fit our system.

A. Temperature Measurement

From our previous design, we decided to change our temperature measuring hardware from a FLIR IR camera to a MLX90614_DCI medical grade long distance IR sensor. We made this change since IR cameras within our price range had around 2 degrees of error, which is significant when referring to human body temperatures. More accurate IR cameras ranged from around $1000 to $2000, which is out of our budge. The IR sensor in contrast measures temperature with an accuracy of ±0.2°C. It measures the surface temperature by detecting infrared radiation energy and wavelength distribution. The sensor can detect object temperatures from a range of up to 50cm, which still conforms to our project goals of ensuring safe social distancing temperature measurement, since our kiosk does not require human operation.

B. Serial Communication Protocol

We also considered which serial communication protocol would be best suited for sending RFID tag ids from the Nucleo to the Jetson. The two that were under consideration were I2C and SPI. We ended up choosing I2C over SPI because it supports multiple masters to multiple slaves on the bus, making the project more scalable. In addition, I2C has ACK and NACK bits to support error handling, which the SPI protocol does not have. While SPI is better for shorter distances and I2C is meant for long distance communication, our primary form of identification is done through the RFID tag ids. Thus, error handling is necessary to prevent inaccurate storage of information and ensure that the correct identification is stored in our local and cloud systems.

C. Microcontroller Unit

We originally planned on using an Arduino, Raspberry Pi, or FPGA as our main processor for the RFID verification process. We ruled out the FPGA early on because the functionality we wished to achieve was not able to utilize the full capabilities of an FPGA. From getting some feedback from the Teaching Assistants, we pivoted towards using an MCU. The first board that came to mind was to use an Arduino since we have previously worked with this microcontroller in past classes. We also considered using a Raspberry Pi but ended up ruling that out. This is discussed in the next section. In addition, Arduino and Raspberry Pi boards are more for experimenting instead of creating an end-to-end prototype. We wanted to target our product to be in the industrial setting, and thus we ended up moving towards using an embedded board. The STM32 series seemed promising since it seemed to be a popular choice with various expansion boards. We wanted to aim for a low power series that would also be compatible with an NFC/RFID expansion board, so we ended up choosing the STM32 Nucleo L476RG.

D. Jetson Nano

We chose the Jetson Nano as our core processor over the Raspberry Pi because of its higher performing GPU. We determined that the Jetson Nano would be the better choice since it could process around 15 frames per second versus the Raspberry Pi's 1 frame per second. With the higher frame processing speed, we would be able to recognize the face and mask faster, and be able to match our goal of scanning a person's temperature at a rate of 3 seconds in order to compete with handheld thermometers.

V. System Description

The project consists of a software and hardware component. On the hardware side, a microcontroller unit (MCU) was used to handle interactions with a RFID tag. The Jetson processes a video stream for facial and mask detection. This information will then be sent to the cloud using Azure.

A. STM32 Nucleo Board

The STM32CubeIDE is used to compile and download code to the microcontroller. The expansion board provided the drivers for X-CUBE-NFC02 and M24LR. However, the provided code was only compatible with F401X and L01X STM32 boards. Necessary changes and modifications were made to ensure that the drivers worked with our L4XX series board. Standard HAL drivers were downloaded that were specific to our board. The HAL library is used to provide generic APIs, such as initializing and configuring the necessary peripherals, managing data transfers, and managing communication errors. A middleware folder with NDEF libraries was also provided. This served as the main documentation for working with NFC type 5 tags. A tag interface is written to check whether the id already exists within our system. Then an I2C interface sends the tag id to the Jetson nano for further processing.

To prevent Jetson from being automatically powered on when connected to a power supply, we are shorting pins 5 and 6 on the J50 header of the Jetson to disable auto-power on. A "wake up" signal is sent from the Nucleo to the Jetson by shorting pins 11 and 12 of the J50. Thus, the Nucleo and the Jetson are directly wired. Pins 3 and 5 on the J41 header is used to connect the SDA and SCL wires needed for our I2C serial communication protocol. We are powering the Jetson through the barrel jack using 5V 4A so J48 is jumpered to enable this feature.

To perform our facial and mask detection, we are using the Raspberry Pi camera module V2 as our main video feed. This is directly connected to the Jetson through J13 for camera connector #1. The Raspberry Pi camera module V2 communicates with the Jetson Nano through the MIPI CSI-2 interface. It is a high-speed protocol used for point to point image and video transmission between cameras and host devices. We also installed a Wi-Fi card, Intel Dual Band Wireless-AC 8265 and Wi-Fi antennas onto the Jetson Nano for mobility instead of needing to tether it to a router switch. We can then connect the Jetson Nano to the cloud for our IoT platform. All video stream is broadcast back to the user through a display monitor that is connected by an HDMI to HDMI cable on J6. The display monitor has a bounding box to confirm the face of interest, the measured temperature, and other necessary information. The IR temperature sensor, MLX90614_DCI, is connected to the Jetson through one of the GPIO pins on J41. The sensor uses the I2C interface to communicate, so we are connecting the MLX90614_DCI SDI and SCK to the SDA and SCL pins on the Jetson Nano.

B. Jetson Nano

Jetson Nano serves as the software processor. It provides two main purposes — (1) face and mask detection and (2) IoT connection.

The Jetson receives the video stream inputs from the RGB camera and run the facial and mask detection algorithm on each frame. The detection algorithm is implemented in Python using popular computer vision libraries such as OpenCV and NumPy.

For the IoT functionality, we are deploying the IoT Edge Module directly on the Nano. This allows the device to easily connect to Azure IoT hub, which is the cloud gateway that is compatible with IoT Edge devices. We are using the MQTT

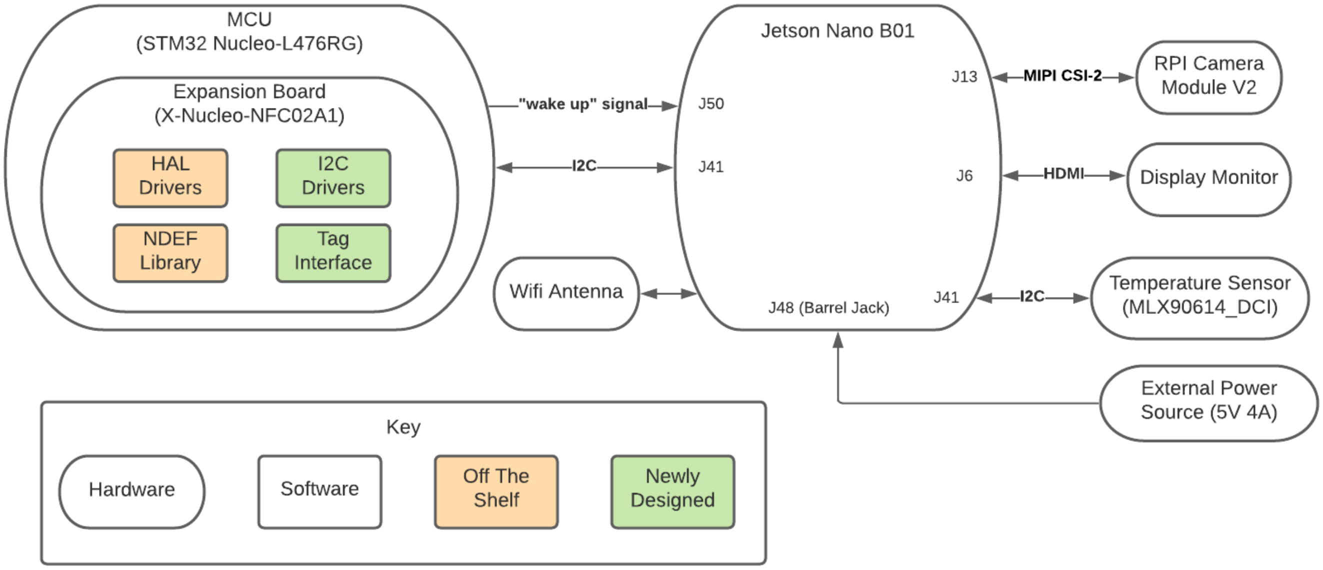


Fig. 4. This is the hardware block diagram for the MCU and Jetson Nano.

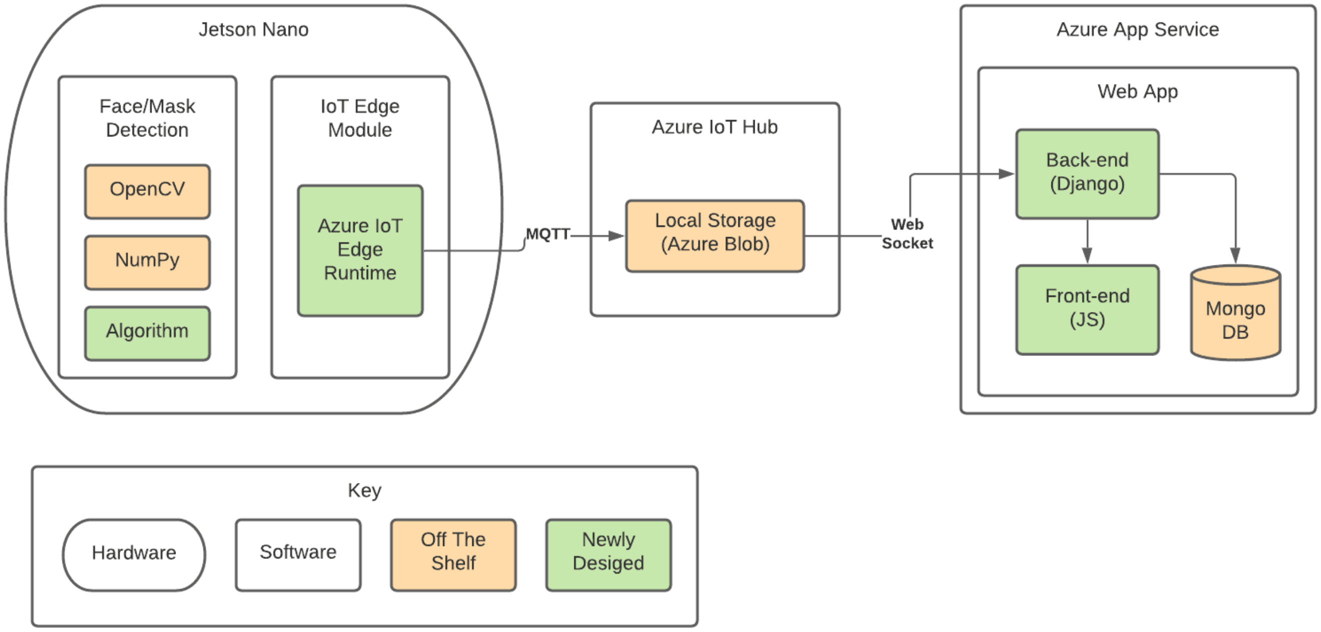


Fig. 5. This is the software block diagram for the Jetson Nano and IoT.

protocol to send messages to the IoT Hub. After we receive the messages, the Hub stores the information in a local storage, Azure Blob, to allow periodic bulk updates in case of internet connectivity issues. Then, the IoT Hub packs the data and sends it over to our Web Application's back-end service. The back-end communicates with the database and the front-end of the application to manage and display the collected temperature data. The back-end service is implemented using Django, while the front-end is built on the React.js framework. Finally, we are using Azure App Service to deploy our web application, since it easily integrates with the IoT Hub and provides a safe way to send and receive data through web sockets.

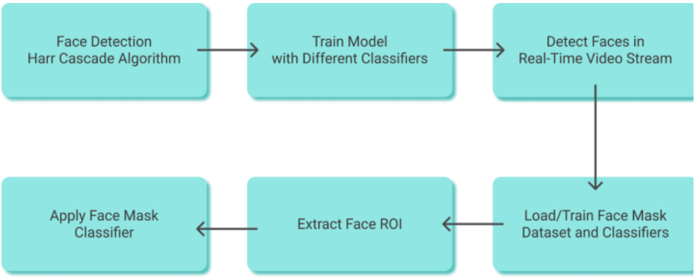


Fig. 6. Facial and Mask Detection Algorithm Flow Chart.

We are using a two-step facial detection algorithm, where we first try to locate a face, and once we have, detect if there is a mask on that face. The face detection model is trained with eye and eyebrow classifiers since we are unable to use nose and mouth classifiers due to the face mask covering up most of those features. Once we compute the bounding box of the face, we apply the mask detection algorithm. We must identify the correct face in the video stream since there may be several people in the background. Thus, there is a bounding box size threshold, and once it passes that, we can identify that as the person of interest. The mask algorithm is trained with a dataset of faces with mask and no mask and may need to supplement it with our own training sets if accuracy is below what we expect. Once the detection algorithm is applied, we can display the mask results and temperature on the monitor, and send temperature, RFID, and other necessary data to the cloud.

VI. Project Management

A. Schedule

The Gantt chart is split up in three different sections: initial set up, MVP, and final project (Appendix Fig. A1). During initial set up, we focused on getting familiar with the hardware and the software packages that are available to us. This is also the time when we submitted our bill of materials and were just waiting on shipment. Due to shipment delays and placing incorrect orders, our Gantt chart did shift back by a couple days, but this did not affect our work significantly.

Next is MVP, these are the tasks that we aimed to finish by the first demo. During this process, we must validate each of the components or code we write before trying to integrate everything together.

In the final project, we continued to push our MVP to the best it can do, such as better accuracy and power efficiency. During this time, we might also explore new areas or components that we wish to add.

B. Team Member Responsibilities

Jiamin focused on working with the RFID scanner and Nucleo board. She made sure that the RFID scanner is properly connected to the Nucleo board, so that the scanned ID is eventually communicated to the Jetson Nano. In addition, she wired the Nucleo to short pins on the Jetson to signal it to turn on or off.

Iris focused on the Jetson Nano board. She made sure that the camera modules are fully integrated with the Jetson and that the facial and mask detection algorithms reached our required accuracy. She also handled the display monitor so that the information recorded by the Jetson is broadcast back to the user.

Minji focused on the cloud and IoT side. She handled the communication from Jetson to cloud, more specifically the encryption and decryption of the messages. She made sure that the messages are sent correctly and at a reasonable speed. With all the user profile information, she developed a web application for accessibility of information to the users.

C. Budget

Table 1 presents all the materials that were purchased for this project. We were given a budget of $600.00 and we have used just about half of the amount.

TABLE I. BILL OF MATERIALS

Component	**Purchase Details**			
	Supplier	***Cost (per unit)***	***Quantity***	***Cost***
Jetson Nano Developer Kit	Amazon	99.00	1	99.00
Jetson Nano Wi-Fi Antenna	Amazon	23.57	1	23.57
Micro SD Card (64GB)	Amazon	11.99	1	11.99
SD Card Reader	Amazon	12.99	1	12.99
RPi Camera Module V2	Amazon	20.00	1	20.00
IR Sensor MLX90614_DCI	Mouser Electronics	60.00	1	60.00
X-NUCLEO-NFC02A1 RFID Scanner	STM Electronics	9.19	1	9.19
STM32 NUCLEO-L476RG	STM Electronics	14.31	1	14.31
ST25-TAG-BAG-A	Mouser Electronics	3.75	1	3.75
ST25-TAG-BAG-U	Mouser Electronics	12.50	1	12.50
Jumper Pins	Amazon	6.99	1	6.99
Wires	Amazon	8.49	1	8.49
HDMI Cable	Amazon	7.00	1	7.00
Total				**299.77**

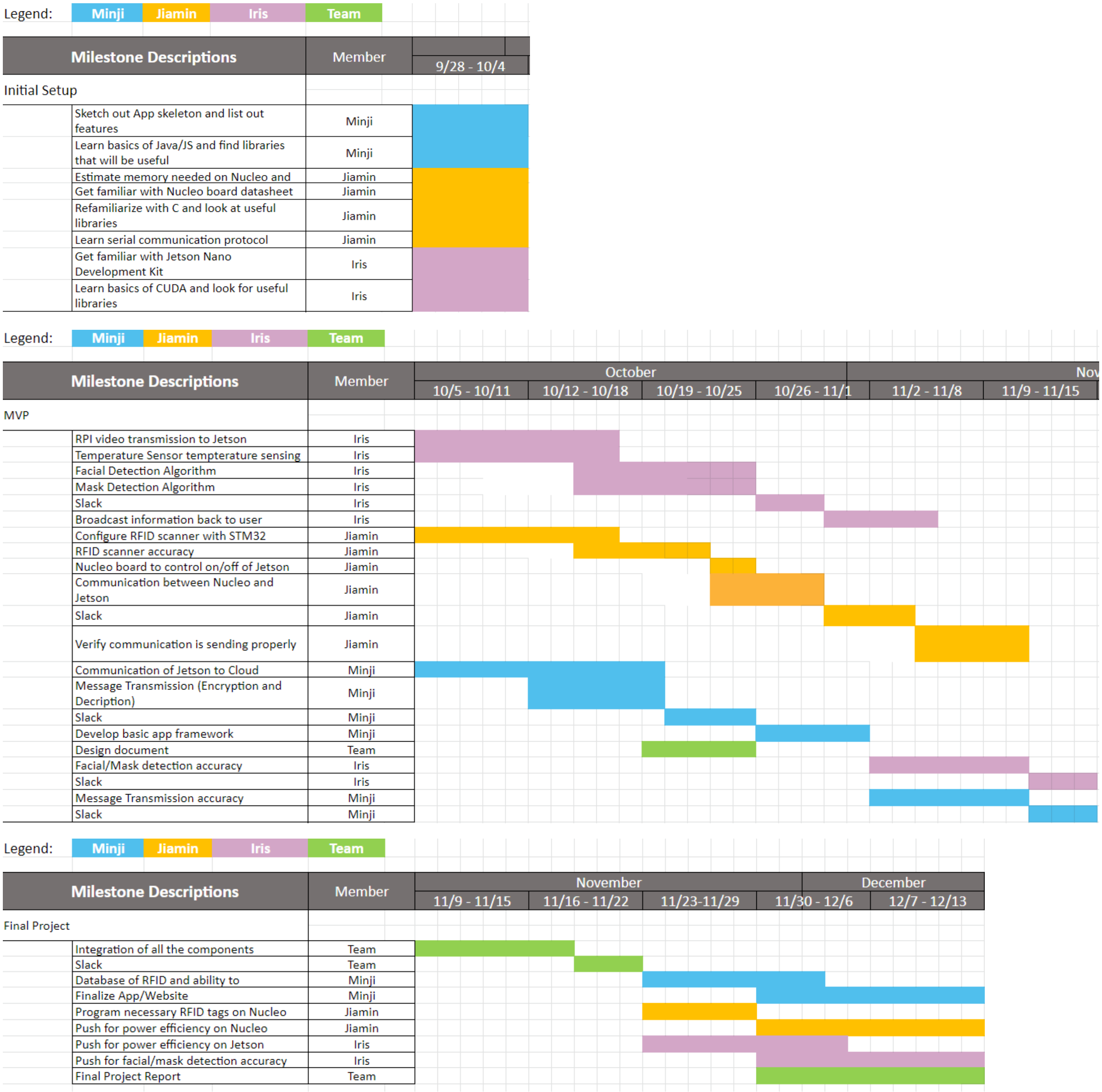
D. Risk Management

We have identified several risk factors for our project. The first one pertains to the accuracy of the mask detection algorithm. Since there will be varying forms of masks, we may have higher false positives and false negatives when detecting masked faces. To address this issue, we are planning on collecting more training sets with a variety of masks.

We also anticipate that the IR sensor may detect inaccurate temperatures. To ensure the accuracy of the temperature data we collect, we will be increasing the sample time to collect multiple data and take the average of them for the final result.

Regarding the IoT aspect of the project, it is also possible that packet loss will occur during the transmission of data between the Nano and the IoT Hub, or between IoT Hub and the web application. This could be due to a lost internet connection, which is a possibility in a real-life application of our system. To account for this situation, we are planning on locally storing recent data in Jetson Nano’s memory and IoT Hub’s local memory and performing periodic bulk updates.

VII. Appendix



Legend: Minji Jiamin Iris Team

Milestone Descriptions		Member	9/28 - 10/4
Initial Setup			
	Sketch out App skeleton and list out features	Minji	■
	Learn basics of Java/JS and find libraries that will be useful	Minji	■
	Estimate memory needed on Nucleo and	Jiamin	■
	Get familiar with Nucleo board datasheet	Jiamin	■
	Refamiliarize with C and look at useful libraries	Jiamin	■
	Learn serial communication protocol	Jiamin	■
	Get familiar with Jetson Nano Development Kit	Iris	■
	Learn basics of CUDA and look for useful libraries	Iris	■

Legend: Minji Jiamin Iris Team

Milestone Descriptions		Member	October				Nov	
			10/5 - 10/11	10/12 - 10/18	10/19 - 10/25	10/26 - 11/1	11/2 - 11/8	11/9 - 11/15
MVP								
	RPI video transmission to Jetson	Iris	■	■				
	Temperature Sensor tempterature sensing	Iris	■	■				
	Facial Detection Algorithm	Iris		■	■			
	Mask Detection Algorithm	Iris		■	■			
	Slack	Iris				■		
	Broadcast information back to user	Iris				■	■	
	Configure RFID scanner with STM32	Jiamin	■	■				
	RFID scanner accuracy	Jiamin		■	■			
	Nucleo board to control on/off of Jetson	Jiamin			■			
	Communication between Nucleo and Jetson	Jiamin			■	■		
	Slack	Jiamin				■	■	
	Verify communication is sending properly	Jiamin					■	■
	Communication of Jetson to Cloud	Minji	■	■	■			
	Message Transmission (Encryption and Decription)	Minji		■	■			
	Slack	Minji			■			
	Develop basic app framework	Minji				■		
	Design document	Team			■			
	Facial/Mask detection accuracy	Iris					■	■
	Slack	Iris						■
	Message Transmission accuracy	Minji					■	■
	Slack	Minji						■

Legend: Minji Jiamin Iris Team

Milestone Descriptions		Member	November			December	
			11/9 - 11/15	11/16 - 11/22	11/23-11/29	11/30 - 12/6	12/7 - 12/13
Final Project							
	Integration of all the components	Team	■	■			
	Slack	Team		■			
	Database of RFID and ability to	Minji			■	■	
	Finalize App/Website	Minji				■	■
	Program necessary RFID tags on Nucleo	Jiamin			■		
	Push for power efficiency on Nucleo	Jiamin				■	■
	Push for power efficiency on Jetson	Iris			■	■	
	Push for facial/mask detection accuracy	Iris				■	■
	Final Project Report	Team				■	■

Fig. A1. Gantt Chart