ECE 558 Final Project Report

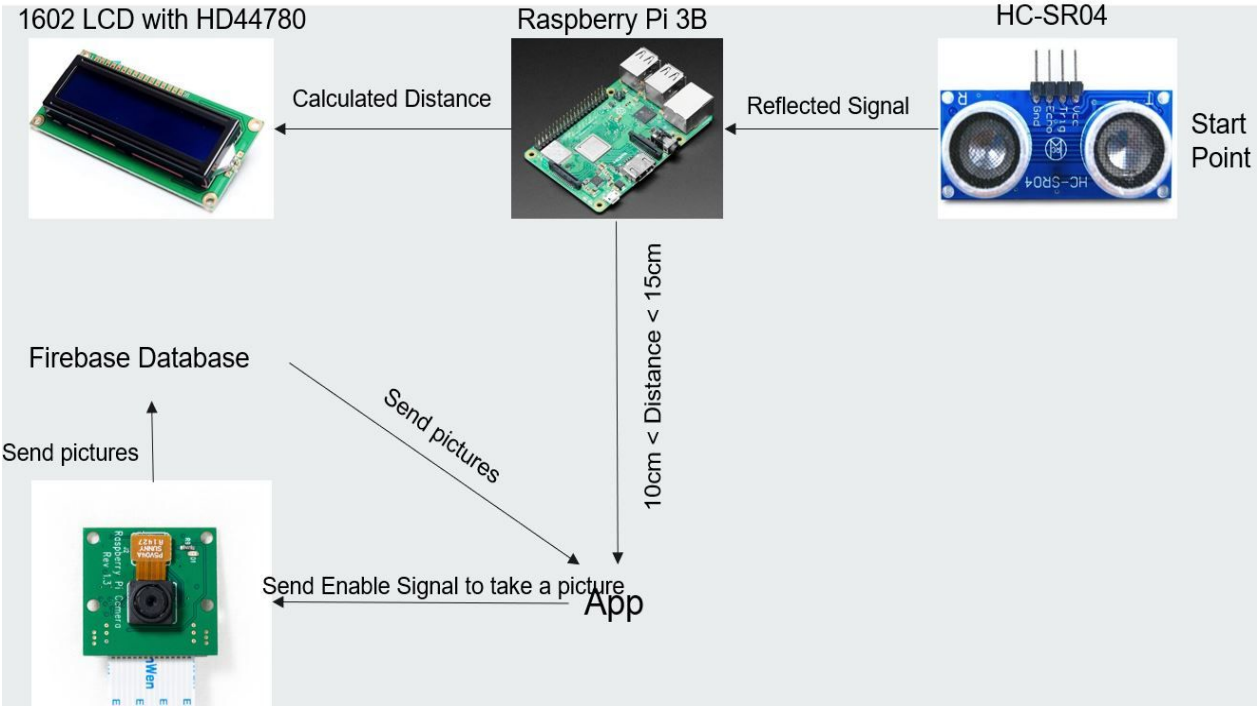
Distance Alert System

Zhe Lu
Ming Ma

Project Description

For this project, we design a Distance Alert System. It consists of a Distance Sensor (HC-SR04) , a LCD display(1602 with build in HD44780 controller), Raspberry Pi 3b board, Raspberry Pi Camera, Firebase database and a user app. The distance sensor will send the signal to Raspberry Pi board, then this board will calculate distance and send the distance to LCD display. When the distance is between 10cm and 15cm, Raspberry Pi board will send an enable signal to app. At this time, app will show a alert dialog to make the user to make a choice: if they want to take a picture. App will send an enable signal to Raspberry Pi camera to take a picture if the user click on yes in the dialog. The captured picture will be send to Firebase database and then send to app from Firebase database. Also, there is a button in the app to clear all photos in app and firebase database to release memory.

Project Transaction Diagram



Design Details

Distance Sensor

For this project, we use HC-SR04 distance sensor. There are 4 pins in this kind of distance sensor. The Vcc and Gnd will connect to 5V and ground of Raspberry Pi respectively. The Trig is output and is connected to GPIO4 of Pi. The Echo is input and is connected to GPIO17 of Pi. However, we need to add a voltage divider between the Echo pin and GPIO17 because the common GPIOs in Pi can only receive input voltage which is smaller or equal to 3.3 Volt. So, I put a 470 ohms resistor and a 330 ohms resistor in serial. Then, the input voltage after using voltage divider is about 3 Volts which is a safe voltage to use. Also, we initially make both Trig and Echo to low. In order to make distance sensor work properly, we need to hold Trig high a least for 10 us. Then the Trig pin will keep sending 8 burst of signals. These signals will be reflected when they touch an object and reflected signals will be received by Echo pin. The level of Echo pin will start to become high when Echo receives the signals. And it will become low again when Echo pin finish receiving reflected signals. The pulse width of Echo becoming high is proportional to the distance that we want to calculate. The equation that I used to calculate distance will be discussed in Equation part.

LCD Display

For the LCD display part, I use 1602 LCD display with build in HD44780 controller. When I start to design this part, I want to use PI4J library to finish this part. However, the PI4J Library cannot statically link to WiringPi library which is configured as common connections between Raspberry Pi board and this kind of LCD display. I try lots of times for this part, but I can't figure out how to dynamically link WiringPi. So, I search online and I find another reference library which is [1]. This LCD has two rows and there are 16 columns in each row. I put the calculated distance in row 0 and I put the current time in row 1. However, I broke one LCD display because I make a mistake that I connect both Vdd and ground to 5V source. So, the LCD is broken and I buy a new one. The left parts for LCD is pretty smooth.

Raspberry Pi Camera

For the camera part, I use the the Raspberry Pi Camera Module v2 which is specific designed for the Raspberry Pi. This camera does not require any I2c or GPIO pinout so that I do not need to care for the connection of the pin. There is a port in the Raspberry Pi which this camera can use it directly. However, when I tried the camera with this port and run some simple code to test the connection, the Raspberry Pi always shows it cannot be detected which means the connection between the Raspberry Pi and the camera is not stateable and I think there maybe some short between them. Then I tried to fix them in many different ways. For example, there are some dust inside the port or

in the surface of the cable. After I tried so many different ways, I found that the port for the camera in the Raspberry should be put down after the cable connect.

User App

For the User App part, the feature if it is receiving the distance which detected by the sensor and display it by using firebase realtime database, and then if it is in the specific range, like what we setup is between 10 to 15 cm, there will be a alert dialog appears to remind the users that the object is already in that specific range that they are allowed to take an image or just do not do anything for that. If the user click yes which means he wants take the photo, the image will be taken by using the camera and then the user app will get it through the firebase storage. After a few seconds delay which depends on the Internet, the image will show up. Also, if the user click the image, it will zoom in and the image will fill out the screen, then click again it will back to the original version and the location of the image will shift left. Moreover, there is another button that can delete all the image both in the user app and in the firebase.

Equations

$$Distance = (((pulse\ width\ in\ ns)/(10^9)) * 340/2) * 100$$

As shown in the equation above, we get the pulse width(high signal in ns) for Echo pin. Then we divide that by 10^9 to convert from ns to s. Next, this value multiples the speed of sound in air which is 340 m/s to get the distance in m. Then, divide this value by 2 because there is a reflection process. Finally, we multiply this by 100 to convert the unit of distance from meter to centimeter. We use this equation in this project to calculate the distance between the distance sensor and an object.

Important Code Explanation

Distance Sensor

```
mTrigger.setValue(false);
Thread.sleep( millis: 0,  nanos: 2000);
//Hold the trigger High for at least 10 us
mTrigger.setValue(true);
Thread.sleep( millis: 0,  nanos: 10000); //10 us
mTrigger.setValue(false);
```

As shown above, this code set Trig to low initially. Then, hold Trig high for 10 us. After this, make Trig to low again. This will make distance sensor keep sending 8 burst of signals.

```
//Reset the trigger pin
while(mEcho.getValue() == false) {
    mReceiving = 0; //make this while loop keep working
}
long mTimeStart = System.nanoTime();
Log.i(TAG,  msg: "Echo Arrived.");

//Wait for the end of the pulse on the ECHO pin
while(mEcho.getValue() == true) {
    mReceiving = 1;
}
long mTimeEnd = System.nanoTime();
Log.i(TAG,  msg: "Echo Ended.");

long pulseWidth = mTimeEnd - mTimeStart;
```

As shown above, the time(mTimeStart) will be recorded when the Echo is starting to become high which means it start to detect object. Then, another time(mTimeEnd) will be recorded when the Echo is finishing keeping high which means it just has finished detecting object.

LCD Display

```
lcd = new Lcd1602(GPIO_LCD_RS, GPIO_LCD_EN, GPIO_LCD_D4, GPIO_LCD_D5, GPIO_LCD_D6, GPIO_LCD_D7);
//The LCD has two rows and each row has 16 columns.
lcd.begin( cols: 16,  lines: 2);
```

As shown above, I use the Lcd1602 library and declare the port connections. The second command means that this LCD has 2 rows and has 16 columns for each row.

```
lcd.clear(); //clear LCD display before writing new data
lcd.print("Distance:" + mDistance + "cm"); //Write calculated distance to LCD display
lcd.setCursor( col: 0,  row: 1); //Move the cursor to row 1 and column 0. Ready for writing data to row 1.
lcd.print(getCurrentTime());  //Write the current time to row 1.
```

We need to clear LCD display first when we want to write new data to the LCD display. Then, I write the calculated distance to the first row of LCD display. Then, move the cursor to the row 1 and column 0. Besides, I write current time to the second row.

Raspberry Pi Camera

```
private void onPictureTaken(final byte[] imageBytes) {
    if (imageBytes != null) {
        final DatabaseReference log = mDatabase.getReference("logs").push();
        final StorageReference imageRef = mStorage.getReference().child("images").child(log.getKey());
        // upload image to storage
        UploadTask uploadTask = imageRef.putBytes(imageBytes);
        Task<Uri> urlTask = uploadTask.continueWithTask(new Continuation<UploadTask.TaskSnapshot, Task<Uri>>() {
            @Override
            public Task<Uri> then(@NonNull Task<UploadTask.TaskSnapshot> task) throws Exception {
                if (!task.isSuccessful()) {
                    throw task.getException();
                }

                // Continue with the task to get the download URL
                return imageRef.getDownloadUrl();
            }
        }).addOnCompleteListener(new OnCompleteListener<Uri>() {
            @Override
            public void onComplete(@NonNull Task<Uri> task) {
                if (task.isSuccessful()) {
                    Uri downloadUri = task.getResult();
                    Log.i(TAG, "Image upload successful");
                    log.child("timestamp").setValue(ServerValue.TIMESTAMP);
                    log.child("image").setValue(downloadUri.toString());
                } else {
                    // clean up this entry
                    Log.w(TAG, "Unable to upload image to Firebase");
                    log.removeValue();
                }
            }
        });
```

This method is the main method for the camera part. After the camera get the 320x240 full image, the image should be stored into the firebase database. The way how it works is first check the image byte is there or not, if there is no the image what we just take, it will upload the image into the storage by using the url which store inside the firebase realtime database. You can see above, there is a firebase child which called “logs” are used to stored the url. After the firebase storage get that ult, it can upload the image into it.

User App

For the app part, I used the recycleview to display the image which download from the firebase storage.

```
// Display the image
if (model.getImage() != null) {
    StorageReference imageRef = mFirebaseStorage.getReferenceFromUrl(model.getImage());

    GlideApp.with(mApplicationContext)
            .load(imageRef)
            .placeholder(R.drawable.ic_image)
            .into(holder.image);
}
```

The code above is used to display the image in the recycle view. I override a onBindViewHolder method to hold the view to display the image.

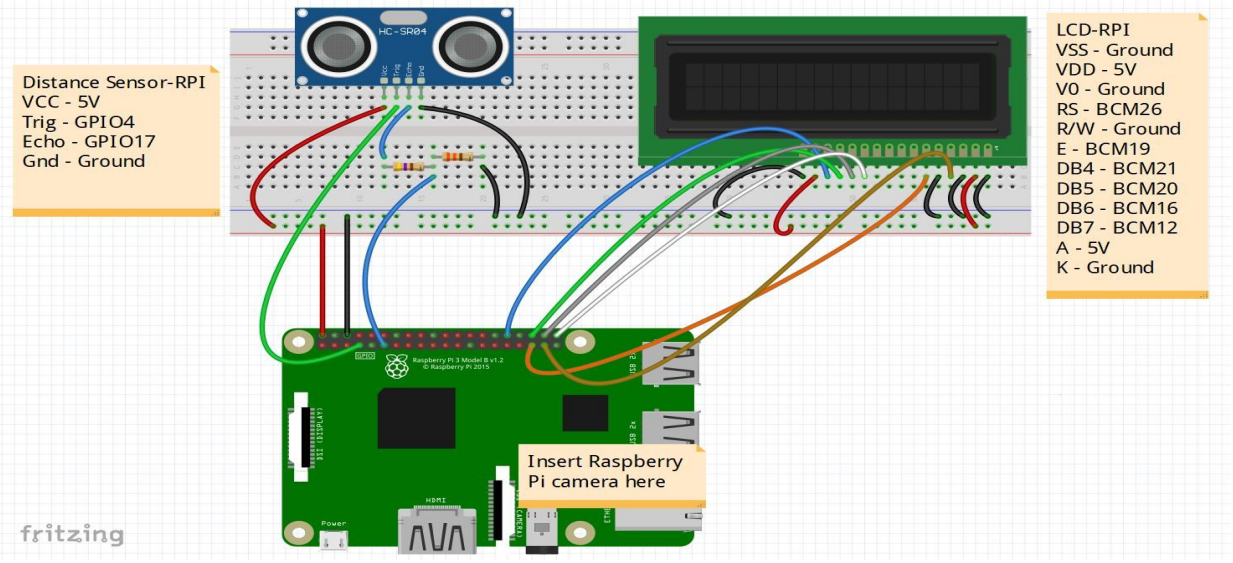
```
image.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (isImageFitToScreen) {
            isImageFitToScreen = false;
            image.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.WRAP_CONTENT, LinearLayout.LayoutPar
            image.setAdjustViewBounds(true);
        } else {
            isImageFitToScreen = true;
            image.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT, LinearLayout.LayoutPar
            image.setScaleType(ImageView.ScaleType.FIT_XY);
        }
```

Also, this method I override is to zoom in the image to the full screen so that the user can check the image they take clearly. I used the setOnClickListener so that every time if the user click the image, it will change the layout which fits the whole screen.

Result

Overall, we finish our goals that we list in the project proposal. The whole project works well. One thing that I want to mention is the Distance sensor part. The distance sensor sends pulses with 15 degree angle. So, we need a big and smooth object to reflect signals. Another point is the minimum distance that this sensor will detect is about 9 cm and the result will become unstable when this distance sensor receives lots of reflected signals. This maybe due to the coding language. We achieve this detecting function by using JAVA. However, the best language is used to achieve this is Python. For the delete button in app, we don't achieve the function of deleting captured pictures in Firebase storage because Firebase doesn't support delete all files at one time. However, we do delete the photos in both app and Firebase database to release memory.

Setup Project



Contribution of Each Member

Ming Ma: I did the distance sensor part and LCD display part.

Zhe Lu: I did the Raspberry Pi Camera and User App part.

Reference

[1].https://hackernoon.com/android-things-basics-measure-distance-with-ultrasonic-sensor-3196fe5d7d7c. Reference design for Distance sensor section.
[2].com.nilhcem.androidthings:driver-lcd1602:0.0.3. Reference Library for LCD display.