Button Controlled Patient to Nurse Communication in Hospitals

Sean A. Hanson
Oregon Institute of Technology
ENGR Capstone Project
Wilsonville OR
sean.hanson@oit.edu

***Abstract*— One of the most frustrating things a patient in their hospital bed face, is pushing the red call button on their bed and having to wait for a nurse to come. This button controlled device addresses this issue. The Amazon Echo is programmed to hear the patient's request. The system then converts the voice into text and is then texted to the nurse. The nurse has an application on his/her phone that filters the texts from which room the patient is from. When the nurse acknowledges the request, a visual light notifies that individual patient. The nurse then has the ability to communicate back to the patient that they have either received, acknowledged, or delivered the request. The application keeps track of texts based on which room they are received, and the application will erase, and reset to a neutral screen once the requested item or service is provided. The project is successful at addressing the issues of connecting to any type of WIFI, and it also addresses the issue of connecting multiple WIFI devices, and the communicating phone to the one WIFI network.**

***Index Terms*—Voice recognition, communication, WIFI device, MIT app inventor, Amazon Echo, Amazon SNS, 8266 NodeMCU**

I. Introduction

THINK back to the last time you were in the hospital as a patient, or friend or family of a patient. How long did it take for the nurse to come when the red button was pushed or when the head nurse was called via the intercom? Now imagine not feeling helpless when the nurse doesn't come in right away. Imagine a way of communicating to the nurse where they know exactly what you need, and you know they got your message and are working on it. This device addresses the need for instant feedback of your needed service. With current technology in hospitals, the nurse doesn't know what the patient needs, and he/she hasn't a way to know the priorities if multiple red lights go on at the same time. This product provides a service where the patient will be happier, and the nurse will have a better way of prioritizing the patients' needs in real time.

As voice recognition technology advances, and tools like the Amazon Echo or Amazon Echo Dot become more embedded in our daily lives, there is a lot of room for improvement for general communication between the patient and their nurse in hospitals. My project would be used like this: The patient needs a pillow, glass of water, ask a question, or needs some kind of help from their nurse. Next the patient pushes a button that activates a tool like the Amazon Echo. The patient speaks their needs to the Amazon Echo, and then their need is transformed into text and shows up on the nurse's phone as a text.

The idea for this stemmed from a conversation I had with my Mom, who is a nurse. She talked about how on a typical 10 or 12 hour shift, she could easily walk over 10 thousand steps, and she said the main reason for having to walk so much is having to go to the patients room, then having to retrieve items the patients requests, then returning to their room with the item. She works at a pretty big hospital that has over 500 beds. Out of those 500 beds, she said this communication device could be utilized in 75 to 90 percent of the rooms. This device wouldn't be used for the highly dependent patient. If a nurse has 15 patients and 10 patients can use this device to speed up the requests for the nurse, she can spend more time with patients that need her the most. Although there are some low volume floors, that have systems that work great for them, there are many areas in hospitals that communications could vastly be improved.

Voice recognition is already in hospitals in many forms. The technology is being used for many different types of applications. One such use is to speed up documenting patient information and data. Another is to be able to communicate from one nurse to another nurse without using hands. One thing all these uses have in common is the need to be more efficient.

As found in the modern health care website, which is a leading business news site, there is an article titled "Nurses turn to speech-recognition software to speed documentation". The use talked about in the article is to speed up the time a nurse takes filling out charts and necessary patient information. [1]. To really show how drastic this technology is needed there was a study published by The Permanente Journal titled, "A 36-Hospital Time and Motion Study: How Do Medical-Surgical Nurses Spend Their Time?" [2]. The study of close to 800 nurses found that more time is spent on documentation then on face to face nurse to patient care, by a wide margin. With a system that utilizes voice recognition, the nurse will spend less time on documentation, sometimes upwards of twice as less, with the same level of accuracy of the notes, which will free their time for other duties.

A more widely known use for a voice recognition devices is

the use for communicating. In an article titled, “The Effects of Hands Free Communication Devices on Clinical Communication: Balancing Communication Access Needs with User Control”, it talks about a study of what nurses really want in a communication device. The study found that nurses wanted voice recognition that would maintain effective communications, while having a sense of social control. One of the bigger issues the nurses brought up was since they have to use voice instead of text, the possibility of speaking out patient confidential information could be an issue. Although there is a risk, the study didn’t find any instances where the patient’s information was actually breached [3].

Another form of voice recognition in hospitals is used in the operating room. From the website Healthcare Business & Technology, the article titled, “Voice-recognition tech could soon be norm in hospitals”, goes into a Boston area hospital where the surgeon during a procedure, talks out load and the system automatically takes and labels pictures of various parts of a patient’s intestinal tract during a colonoscopy [4].

In a magazine titled HFM magazine, there is a piece about a device that give patients more control. It is titled, “Advances in nurse call systems help to put patients in control”. In this article they talk about a company, West-Com Nurse Call Systems Inc. that uses an integrated-solution approach that has an updated version of the red call button. This is on the Novus platform (an interactive service with dynamic search tools) to provide communication among different devices and software applications. This service bundles a lot of different technology and services into one little package. This system can schedule a nurse’s daily routine, so they can proactively meet patient’s needs more effectively. Another thing the system can do is route the patient’s specific needs to different staff. A request for a blanket will go to a nurse’s assistant, while a request for pain meds would go directly to the nurse. Although this device and service doesn’t quite take full advantage of everything voice recognition has to offer, the article goes into future projections. They state, “Future advances will include monitoring all staff activities, planned and unplanned — allowing assessment and monitoring of total staff time required per patient, data analytics will provide predictive information on potential patient risks.” [5].

In the article, “‘Alexa, pull those lab results’: A hospital tries out virtual assistants”, a hospital has an Amazon echo in each of the children’s hospital rooms. One Alexa App that is used in the room explains home instructions the child needs to do once they leave the hospital. Another app mentioned in the article is an Alexa-based app called *KidsMD* that gives parents advice when their children catches a fever, for instance. [6].

With all the new technology emerging around connecting humans, it is just a matter of time before there is company that develops a device and service that revolutionizes the way patients communicate with their nurse, and their overall hospital experience. Although there are many instances of voice recognition already in hospitals, the research comes up short when it comes to using a device like the Amazon Echo to communicate from a patient to their nurse directly. The great news is as hospitals start integrating these new technologies, it will become not a question of if, but a question of when the hospital will implement a service like this project.

II. How it works

Here is how the device would work with a little more detail. In each patient’s room, there is one device. It is a plastic nurse’s hat with Amazon Echo technology. The patient will have a remote control that has the ability to hear the request with a push of a button. The nurse has a phone that has an application they can download. When the application is opened the nurse has the ability to connect the application to all the rooms they are responsible for. Once that is set up they are ready for their work shift. When the patient needs the nurse, the patient pushes the button on the remote, and speaks their need. When the button is activated, the nurse’s hat greets the patient audibly and asks them what the nurse can help them with. There is also a green indicator light on the hat that indicates it’s ready to hear. Now, the patient says out loud what they want. An example would be, “Nurse, please bring me an extra blanket”. The nurse’s hat then repeats the patient’s request and sends the message to the nurse via text. A soft flashing yellow light blinks indicating the message has been sent to the nurse. Out in the hall, the nurse looks down at her phone, and a text message scrolls by, “Room 215 needs an extra blanket”. The nurse then pushes the ‘ACKNOWLEDGE REQUEST” button. The message is still available to look at on her phone, but the yellow flashing light in the patient’s room stops blinking and turns to a solid yellow. Once the nurse gets the extra blanket, delivers it to the room, she pushes the “DELIVERED” button on the app. This indicates the task is closed out. The soft yellow light turns off, and the nurse’s hat waits for the next request.

There are 5 different stages that the communication devices can be in.

1. 

2. 

3.

4.

5.

III. THE DESIGN

Behind the scenes, there are a lot services being utilized all at the same time. Here is graph showing how they communicate with each other. Below that I will explain each part individually and how to set it up.

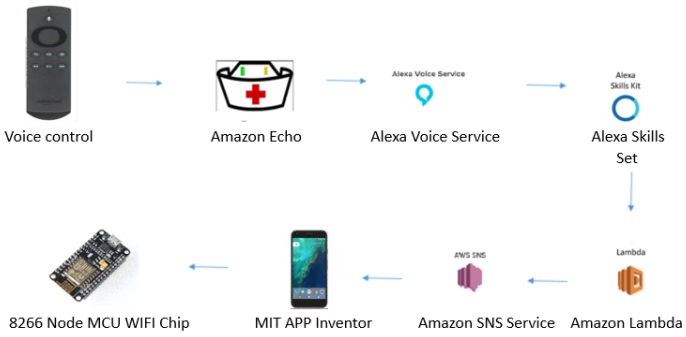


Fig. 1. Complete Diagram of Project

A. Voice Control

The patient pushes the voice control remote, which is synced to the Amazon Echo device via Bluetooth. The patient talks into it and their voice is received by the Amazon Echo via speech to text conversion.

B. Amazon Echo

Once the Amazon Echo converts the speech from the remote control to text, it is listening for keywords. It is activated by the keyword "Nurse". The communication between the remote and the Amazon Echo is then started and the keywords heard by the Echo are used as variables and sent to the Alexa Voice Service.

C. Alexa Voice Service

Open up a browser and go to developer.amazon.com. Sign up for a new account.

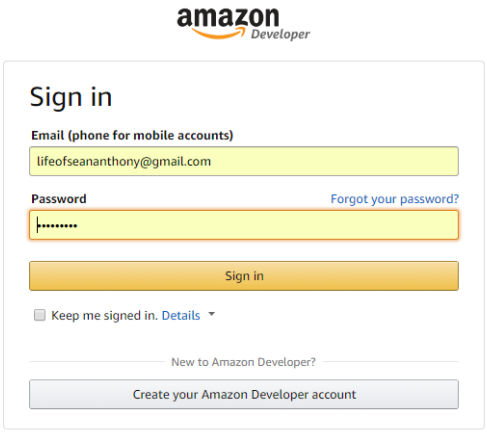


Fig. 2. Screenshot of Amazon Developer log in

Once logged in create a new Alexa Skills Set

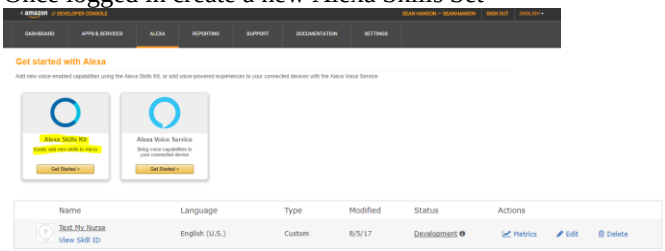


Fig. 3. Screenshot of Amazon Skills under Alexa dropdown

Fill out all the basic information, this includes the name of the skill, what it's used for, and the keyword that will start the skill.

Fig. 4. Screenshot of Amazon Skill Information page

The skills Intention screen is how the Echo interprets what the user says. There are built in words such as 'HELP', 'YES', and 'NO'. For all others the user needs to create them. This is called the Intent Schema.

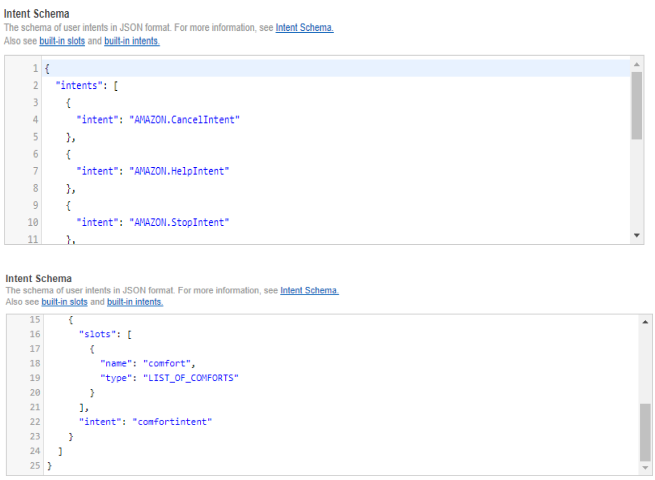


Fig. 5. Screenshot of Intent Schema with built in intents, and the custom slots needed.

The next part are the custom slots and sample utterances. This is an exhaustive list of anything and everything the user could possibly say. The slots will be the variable used for the Amazon Lambda. The sample utterances are what the user could say. You can see the slots are the variable in the utterance section.

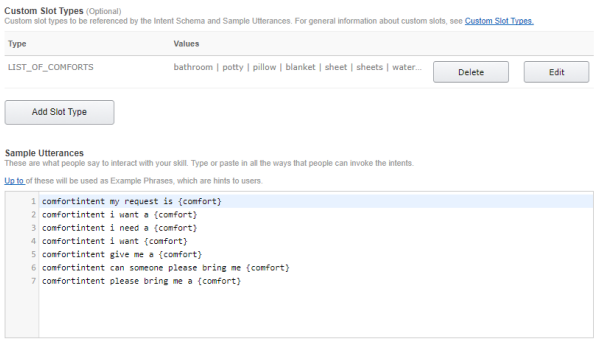


Fig. 6. Screenshot of custom slot, and sample utterances

For the skill to communicate with the Amazon Lambda there is a skill ID. You add this in the configuration tab.

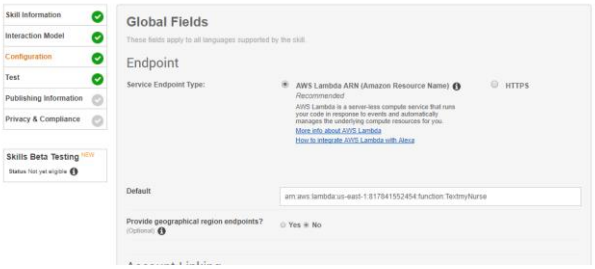


Fig. 7. Screenshot of Configuration tab used to communicate with Amazon Lambda.

The last part of Amazon Skill Set is the Test Tab. Here you get a real time response from the Echo.

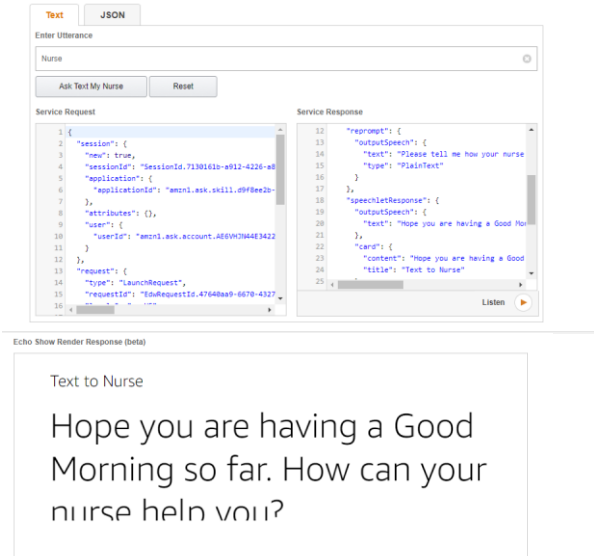


Fig. 8. Actual response when the user says “Nurse”

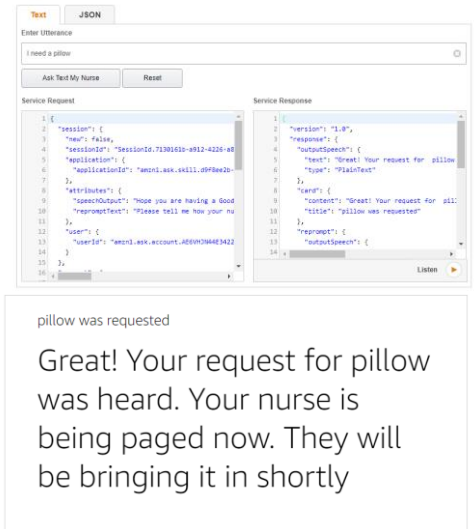


Fig. 9. Actual response when the user says “I need a pillow”

One thing to note here is the variables “Nurse” and “Pillow” are taken from the Skill Set and passed to the Lambda Service. The two above figures show the final test after we set up the Lambda side of things.

D. **Alexa Lambda Service**

Open up a browser and go to *console.aws.amazon.com*.

Sign up with your amazon email and password and open up the lambda tab.

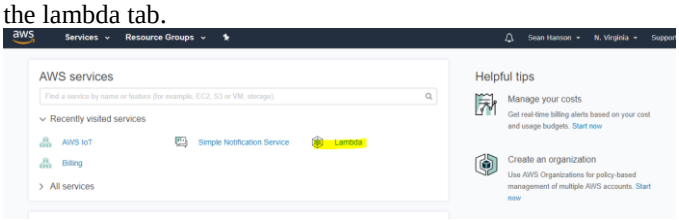


Fig. 10. Signing up for Lambda Service

Create a new function using NodeJS code

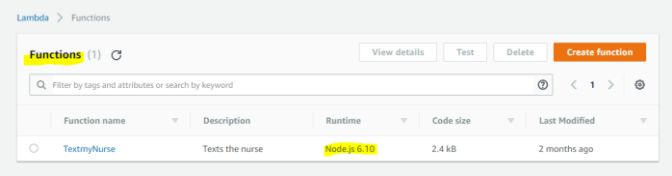


Fig. 11. Creating a new function using NodeJS

In Appendix 1 you will find the complete Lambda code for this project. This is all you need in the lambda service.

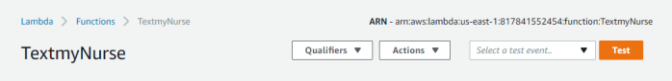


Fig. 12. Created a new function using NodeJS with the ARN

E. **Amazon SNS (Simple Notification Service)**

In the same console.aws.amazon.com log in, look for SNS and create a new service. Here you create the Subscription service that communicates with the phone. First create topics. One for each room in the hospital.

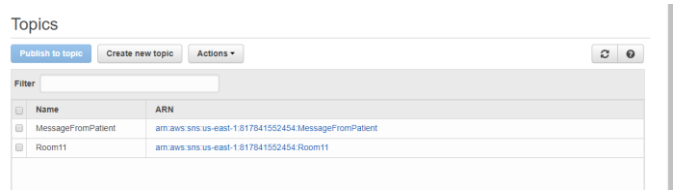


Fig. 13. Created a new subscription for each hospital room

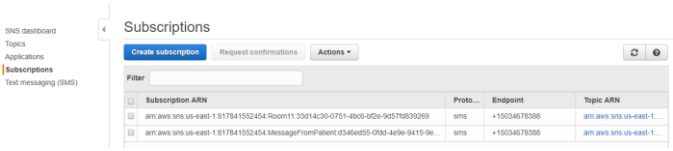


Fig. 14. Created a new subscription for each hospital room attaching the phone number and topic ARN to each subscription.

This will go in the body in the lambda code. Look at Appendix 1, line 140 to see how it is used. Once these subscriptions are set up, that is all you need in the subscription section.

F. **MIT App Inventor**

This is the interface the nurse has between the Echo and the 8266 NodeMCU WIFI chip. To set this up go to http://ai2.appinventor.mit.edu and set up an account.

Fig. 15. Created a new project

The app inventor uses two interfaces per application screen. The designer and blocks. The designer is for the interface of the app, and the blocks are the programming behind the design. There are a lot to this. The designer screens are in Appendix 2. The blocks code is in Appendix 3. The designer and blocks are the only thing in MIT App Inventor. The MIT App Inventor uses puzzle pieces to code. It takes a while to learn how it works but once you get the hang of it, it's just as powerful as any other programming language. As shown in Appendix 3. The block coding can be placed anywhere on the screen. A good reference would be how a Hardware Development Language is used. Everything runs concurrently depending on which button action is selected.

G. **8266 NodeMCU WIFI chip**

To set this up, we use Arduino code. Once it is setup, the chip will automatically connect to the WIFI network via WIFI name and password. The LED is connected to pin3 and controlled by the phone app via the individual IP address assigned to it. There is a pop up for the nurse to enter the given IP address.

```
  // Connect to WiFi network
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  // Start the server
  server.begin();
  Serial.println("Server started");

  // Print the IP address
  Serial.println(WiFi.localIP());
}
```

```
void loop() {
  // Check if a client has connected
  WiFiClient client = server.available();
  if (!client) {
    return;
  }

  // Wait until the client sends some data
  Serial.println("new client");
  while(!client.available()){
    delay(1);
  }

  // Read the first line of the request
  String req = client.readStringUntil('\r');
  Serial.println(req);
  client.flush();

  // Match the request
  int val;
  if (req.indexOf("/gpio/0") != -1)
    val = 0;
  else if (req.indexOf("/gpio/1") != -1)
    val = 1;
  else {
    Serial.println("invalid request");
    client.stop();
    return;
  }

  // Set GPIO2 according to the request
  digitalWrite(2, val);

  client.flush();

  // Prepare the response
  String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\n
  s += (val)?"high":"low";
  s += "</html>\n";

  // Send the response to the client
  client.print(s);
  delay(1);
  Serial.println("Client disonnected");

  // The client will actually be disconnected
  // when the function returns and 'client' object is detroyed
}
```

Fig. 16. Arduino Code for the Node MCU WIFI Chip

The hardware is a pretty simple setup.

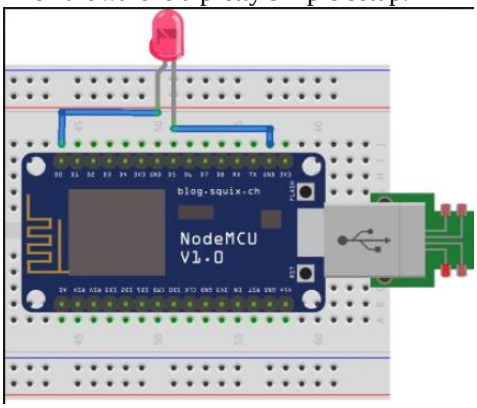


Fig. 17. Node MCU WIFI Chip with connected LED to pin 3

Depending on what buttons are pushed via MIT App Inventor, the LED will be off, on, or blinking. Line 70 is the line that will control if the lot is on or off. In the MIT app inventor, there is a statement that will send a request to the Node MCU which toggles back and forth to on or off. This creates the blink. But there are only 2 states that the chip is in. On or Off.

IV. TESTING AND SETUP

To test this communication system, physical hardware was used. Components that were used to test were An echo dot remote control, echo dot, 2 separate breadboards with one 8266 NodeMCU chip on each board, 2 LED's, and a smartphone with the app on it. All components together can be seen in figure 18. Figure 19 and 20 respectively show the physical components separated for the patient's side, and the nurse's side.

Since the software and hardware are so intertwined with the device, testing was done congruently. Words were spoken into the echo remote, then relayed to the echo dot. Once that was tested, then setting up the speech to text part was done. From there, it was filtering the actual text into text messages, and then have the text and app control the NodeMCU chip and LED. The implementation and testing didn't deviate much at all from the project diagram that's in figure 1. Before moving to the next part of the project, the current part was tested, troubleshot and was working correctly. As the project added more parts, there were slight adjustments along the way. One adjustment was to use WIFI chips instead of Bluetooth chips. Although the Bluetooth chips worked, their range was a problem. Another issue was being able to connect multiple chips to the one device. The WIFI chip connects to the Router which solves the range issue, and since the app connects to the IP address like in Figure 21, of the connected WIFI chip, in theory, the one app could connect to as many chips as the WIFI network would allow. Figure 22 shows what the nurse sees when they click into the blinking room number on the app. This was tested by sending a text via the Amazon Echo, then figuring out how to filter by room number. The mockup of the nurse hat can be seen in Figure 23. A cardboard box was used.

There is just one issue that needs to be resolved. Right now, the system can handle as many rooms and IP addresses necessary. A problem occurs when not all rooms are selected. The app still looks for the IP address of unused chips. The

device still works, just a pop up screen tells the user that the IP address cannot be found.

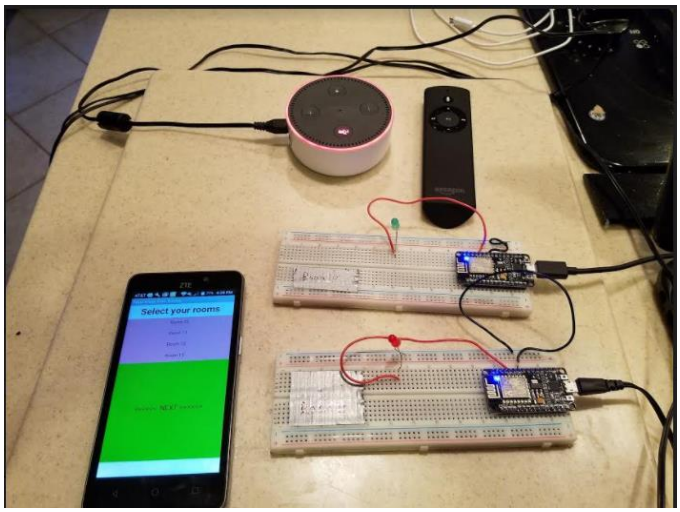


Fig 18. All hardware components


Fig 19. Components used in the patient's room

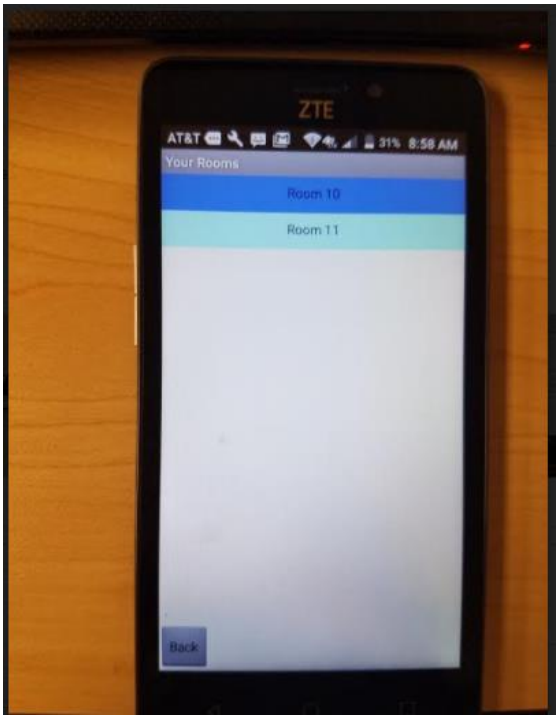


Fig 20. Components used on the nurse's side

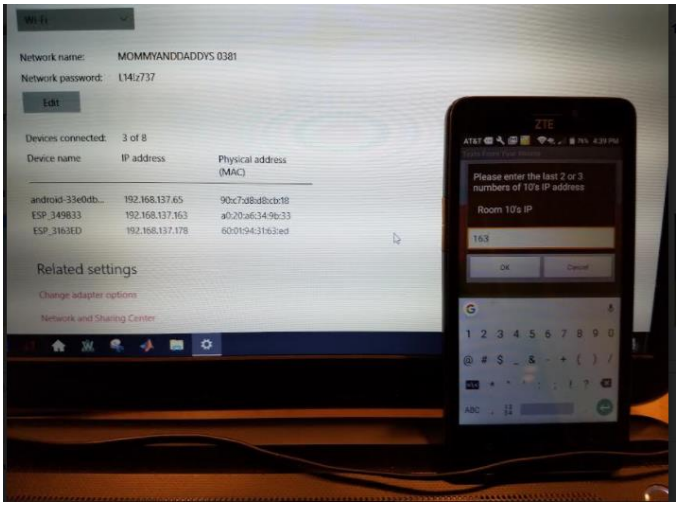


Fig 21. App connecting to each room via IP address

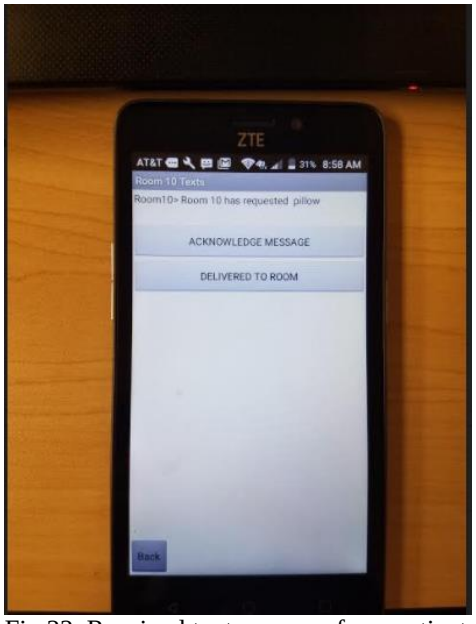


Fig 22. Received text message from patient


Fig 23. Mockup of Nurse Hat

V. COST ANALYSIS

The cost of the project is as follows

Hardware

Amazon Echo Dot (1) ……**$50**
Smartphone (1)……..…**$60**
Phone bill (6 months service).....**$360**
8266 NodeMCU Chip (2)……...**$16**

Software

Amazon Alexa Service……..**Free**
Amazon SNS Service……**$2.50** total. Service is on a pay by text basis. I have sent over 300 texts testing it.
Amazon Skills Service……..**Free**
Arduino……**Free**
Amazon Lambda Service…….**Free**
MIT App Inventor……**Free**

Total cost of project including 6 months of cellphone bills
$ 488.50

VI. POST PROJECT EVALUATION

As I started the project I had the idea of using a smart watch. That proved to be too hard, and once I got feedback from actual nurses, I changed it to a smartphone app. This allowed more flexibility with different WIFI networks. The other change from the original plan was I just used an Amazon Echo and remote instead of building one with a Raspberry Pi. That set up wasn't as responsive as I hoped. The Raspberry Pi Amazon Echo wouldn't hear commands more than half the time. As for everything else, I was pleasantly surprised how the project came along.

VII. CONCLUSION

The communication between a patient and their nurse in a hospital setting can be improved dramatically from where it is currently. This project demonstrates an effective way to communicate using new technologies. Talking into the Amazon Alexa, then using custom skills, the data is sent to an SNS service that sends a text to a smartphone. The app communicates to a WIFI in the patient's room that communicates if the Nurse received or acknowledges the message. This real time communication loop will reduce confusion, patient frustration, and also a nurses footsteps. The best part of this system is, it can be built for less than 75 dollars per room, and easy to use.

To expand on this project, a recommendation would be is, try to get access to the Amazon Echo lights. If one is able to do that, then the service could be inclusive without the NodeMCU chip. The other recommendation would be try to expand the application to include a smart watch. Having that option for hospitals or nursing homes, might excite them to test out this new service.

After concluding the real time usage and feedback from paramedic students, teachers and a nurse in the field, a lot of knowledge about possible changes was gained. Firstly, the Alexa Skill for what the patient can say needs to be a lot more complete. Some other feedback that was given was being able to capture all the messages sent to the nurse. This is something that wasn't thought up in brainstorming. A couple other ideas are to capture sounds instead of words and translate those, and to actually have a device that translates in real time. Being open to the feedback and receiving actual user feedback is invaluable. As the project continues on, and gets closer to being in the hospital, this feedback will help make the design better.

VIII. REFERENCES

1. Conn, J. (December 12th, 2015). 'Nurses turn to speech-recognition software to speed documentation'. http://www.modernhealthcare.com.

2. Hendrich, Ann; Boguslaw A Skierczynski; Zhenqiang Lu,' (Summer 2008) 36-Hospital Time and Motion Study: How Do Medical-Surgical Nurses Spend Their Time?' https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3037121/

3. Richardson, J. E. (2008). 'The Effects of Hands Free Communication Devices on Clinical Communication: Balancing Communication Access Needs with User Control'. https://www.ncbi.nlm.nih.gov

4. White, J. (June 1st, 2016). 'Voice-recognition tech could soon be norm in hospitals.' http://www.healthcarebusinesstech.com

5. Lorenzi, Neal (April 5th, 2017). "Advances in nurse call systems help put patients in control" https://www.hfmmagazine.com/articles/2790-advances-in-nurse-call-systems-help-put-patients-in-control

6. Bailey, Melissa (May 3st, 2016). "'Alexa, pull those lab results': A hospital tries out virtual assistants". https://www.statnews.com/2016/05/31/hospital-virtual-assistants/

APPENDIX 1

COMPLETE LAMBDA CODE

```
/**
 * This Amazon Alexa skill is used for a patient in a hospital to text their nurse
 */
var date = new Date();
var time = date.getHours() ;
var AWS = require('aws-sdk');
// Route the incoming request based on type (LaunchRequest, IntentRequest, etc.)
// The JSON body of the request is provided in the event parameter.

exports.handler = function (event, context) {
    try {
        console.log("event.session.application.applicationId=" + event.session.application.applicationId);

        /**
         * Uncomment this if statement and populate with your skill's application ID to
         * prevent someone else from configuring a skill that sends requests to this function.
         */
        if (event.session.application.applicationId !== "amzn1.ask.skill.d9f8ee2b-628d-4fc2-87ea-d72e68ffcb8d") {
            context.fail("Invalid Application ID");
        }
        if (event.session.new) {
            onSessionStarted({requestId: event.request.requestId}, event.session);
        }
        if (event.request.type === "LaunchRequest") {
            onLaunch(event.request,
                event.session,
                function callback(sessionAttributes, speechletResponse) {
                    context.succeed(buildResponse(sessionAttributes, speechletResponse));
                });
        } else if (event.request.type === "IntentRequest") {
            onIntent(event.request,
                event.session,
                function callback(sessionAttributes, speechletResponse) {
                    context.succeed(buildResponse(sessionAttributes, speechletResponse));
                });
        } else if (event.request.type === "SessionEndedRequest") {
            onSessionEnded(event.request, event.session);
            context.succeed();
        }
    } catch (e) {
        context.fail("Exception: " + e);
    }
};

//Called when the session starts.
function onSessionStarted(sessionStartedRequest, session) {
    console.log("onSessionStarted requestId=" + sessionStartedRequest.requestId +
        ", sessionId=" + session.sessionId);
}

function onLaunch(launchRequest, session, callback) {
    console.log(`onLaunch requestId=${launchRequest.requestId}, sessionId=${session.sessionId}`);
        getWelcomeResponse(time,callback);
}

//Called when the user specifies an intent for this skill.
function onIntent(intentRequest, session, callback) {
    console.log("onIntent requestId=" + intentRequest.requestId + ", sessionId=" + session.sessionId);

    var intent = intentRequest.intent;
    var intentName = intentRequest.intent.name;
    // Dispatch to your skill's intent handlers
    if (intentName == "comfortintent") {
        textintent(intent, session, callback);
    } else if (intentName == "AMAZON.HelpIntent") {
        texthelpintent(intent, callback);
    } else if (intentName == "AMAZON.NoIntent") {
        textnointent(session, callback);
    } else {
        throw "Invalid intent";
    }
}

/**
 * Called when the user ends the session.
 * Is not called when the skill returns shouldEndSession=true.
 */
function onSessionEnded(sessionEndedRequest, session) {
    console.log("onSessionEnded requestId=" + sessionEndedRequest.requestId +
        ", sessionId=" + session.sessionId);
    // Not being used in this skill
}

// --------------- Functions that control the skill's behavior -----------------------
function getWelcomeResponse(time, callback) {
    var timeofday;
        if ( time > 6 && time < 20) {
            timeofday = "Good Morning";
        } else if (time > 20 && time < 23 || (time > 0 && time < 1 )) {
            timeofday = "Good Afternoon";
        } else if ( time > 1 && time < 6) {
            timeofday = "Good Evening";
        }


    var cardTitle = "Text to Nurse";
    var speechOutput = "Hope you are having a " + timeofday + " so far. How can your nurse help you?";
    // If the user either does not reply to the welcome message or says something that is not
    // understood, they will be prompted again with this text.
    var reprompt = "Please tell me how your nurse can help you." +
                   " You can ask. Can I have a pillow? or, I need more medication";
    var shouldEndSession = false;
    var sessionAttributes = {
        "speechOutput" : speechOutput,
        "repromptText" : reprompt
    };
        callback(sessionAttributes, buildSpeechletResponse(cardTitle, speechOutput, reprompt, shouldEndSession));
    }
```

```
/**
 * Intent that text message nurse.
 */
function textintent(intent, session, callback) {
    var whichcomFort = intent.slots.comFort.value;
    var validcomforts = [ "bathroom", "pillow", "blanket", "sheet", "water", "blanket","pills", "meds", "medication"];
    var cardTitle   = whichcomFort + " was requested" ;
    var sessionAttributes = {};
    var shouldEndSession = true;
    var speechOutput = "";
    var repromptText = "";

    if (validcomForts.indexOf(whichcomfort) == -1) {
        speechOutput = "I'm not sure what you are asking for. Please repeat";
        repromptText = "Please say the word help, if you need a nurse immediately";
        callback(sessionAttributes, buildSpeechletResponse(cardTitle, speechOutput, repromptText, shouldEndSession));
    }
    else {
        textyesintent(whichcomfort, function(speechOutput){
            speechOutput = "Great! Your request for  " + whichcomfort + "  was heard. Your nurse is being paged now. They will be bringing it in shortly";
        callback(sessionAttributes, buildSpeechletResponse(cardTitle, speechOutput, repromptText, shouldEndSession));
        });
    }
}
```

```
function textyesintent(whichcomfort, callback){
        var sns = new AWS.SNS(); //Needed in order to send message to Amazon SNS
        sns.publish({
        Message: 'Room 10 has requested  ' + whichcomfort + '',
        TopicArn: 'arn:aws:sns:us-east-1:817841552454:MessageFromPatient' //Add your SNS ARN here
    }, function(err, data) {
        if (err) {
            console.log(err.stack); //an error occured
            callback("Error when sending message");
        }
        console.log('push sent'); //message sent
        console.log(data);
        callback("");
    }); //Logs if your SNS message was sent.

    }
    function texthelpintent(intent, callback){
     var sns = new AWS.SNS(); //Needed in order to send message to Amazon SNS
      var cardTitle = "Patient needs immediate help";
    var speechOutput = "Your nurse has been paged to come in immediately. Please stay in bed until they arrive.";
    // If the user either does not reply to the welcome message or says something that is not
    // understood, they will be prompted again with this text.
    var reprompt = "";
    var shouldEndSession = true;
    var sessionAttributes = {
        "speechOutput" : speechOutput,
        "repromptText" : reprompt
    };
```

```
    sns.publish({
        Message: 'Room 11 is requesting your help immediately',
        TopicArn: 'arn:aws:sns:us-east-1:817841552454:Room11' //Add your SNS ARN here
    }, function(err, data) {
        if (err) {
            console.log(err.stack); //an error occured
            callback("Error when sending message");
        }
        console.log('push sent'); //message sent
        console.log(data);
        callback("");
    }); //Logs if your SNS message was sent.
   callback(sessionAttributes, buildSpeechletResponse(cardTitle, speechOutput, reprompt, shouldEndSession));
    }


function textnointent(session, callback) {
    var speechOutput = "Oops, ok, what is it that you need?";
        var reprompt    = "Are you still thinking? What is it that you need?";
        var cardTitle   = "Wrong item asked for";
        var sessionAttributes = {
        "speechOutput" : speechOutput,
        "repromptText" : reprompt
        };
        var shouldEndSession = false;
        callback(sessionAttributes, buildSpeechletResponse(cardTitle, speechOutput, reprompt, shouldEndSession));
}
```

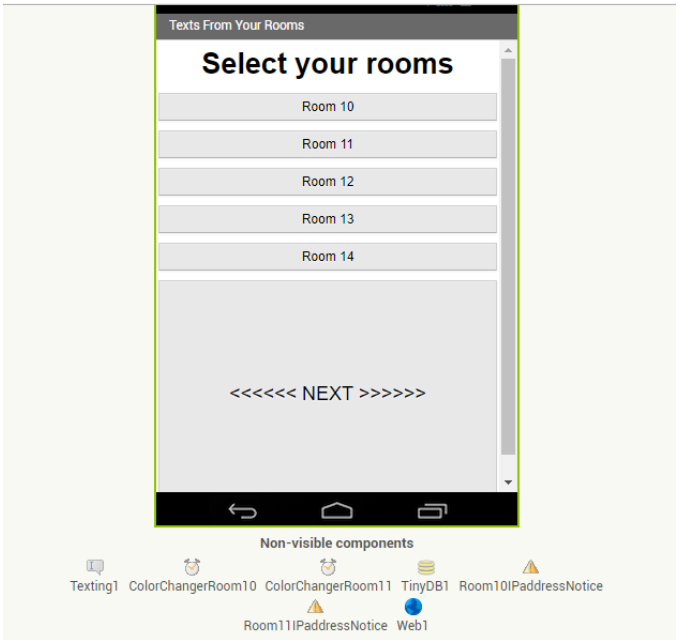
```
// --------------- Helpers that build all of the responses ----------------------
function buildSpeechletResponse(title, output, repromptText, shouldEndSession) {
    return {
        outputSpeech: {
            type: "PlainText",
            text: output
        },
        card: {
            type: "Simple",
            title: title,
            content: output
        },
        reprompt: {
            outputSpeech: {
                type: "PlainText",
                text: repromptText
            }
        },
        shouldEndSession: shouldEndSession
    };
}

function buildResponse(sessionAttributes, speechletResponse) {
    return {
        version: "1.0",
        sessionAttributes: sessionAttributes,
        response: speechletResponse
    };
}
```

APPENDIX 2

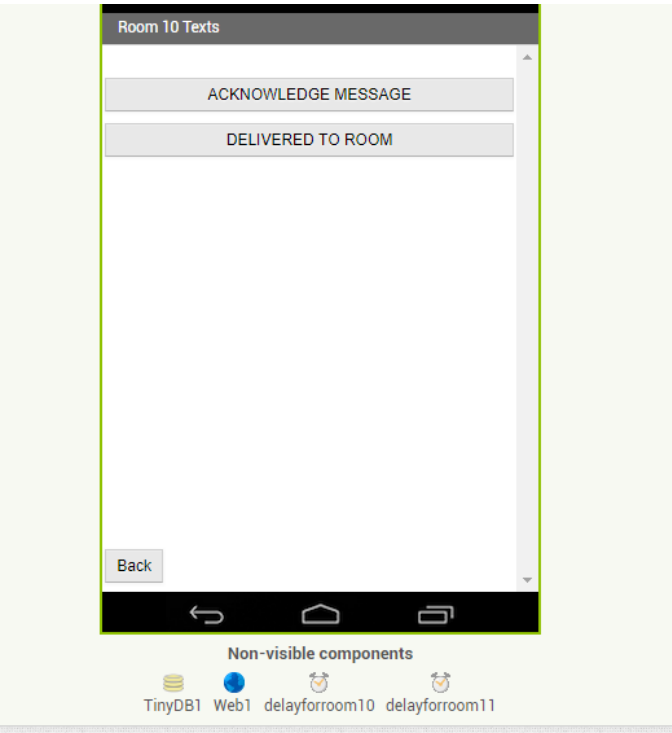
COMPLETE MIT APP INVENTOR DESIGNER CODE

SCREEN 1

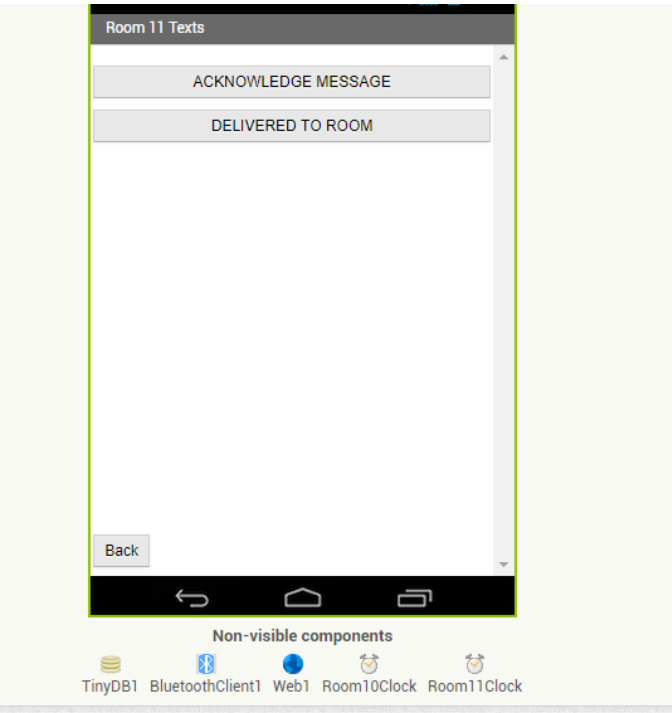


SCREEN 2

SCREEN 3



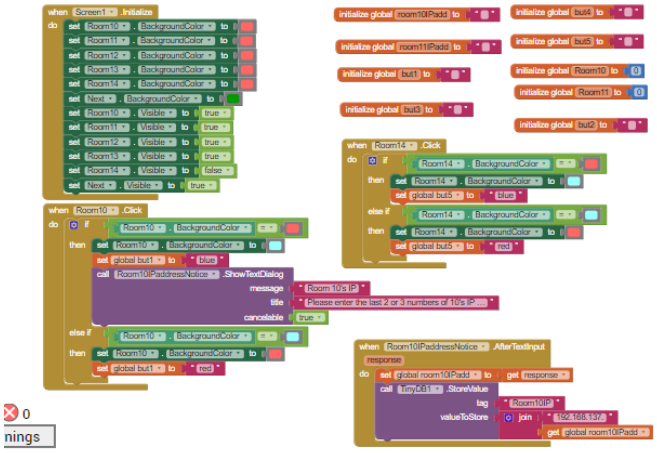
SCREEN 4

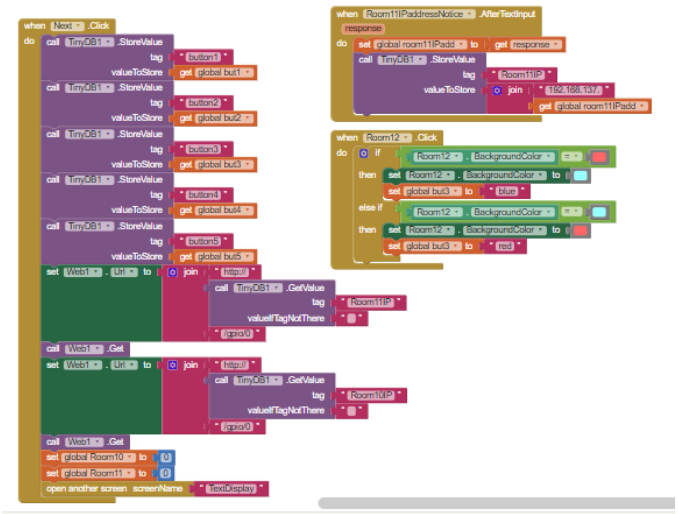


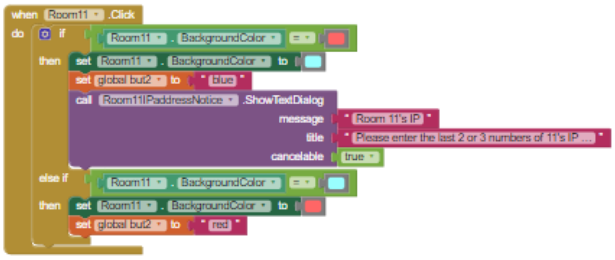
APPENDIX 3

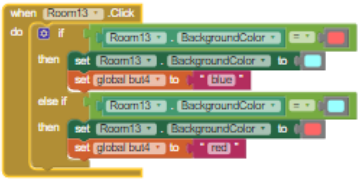
MIT APP INVENTOR BLOCK CODE

SCREEN 1

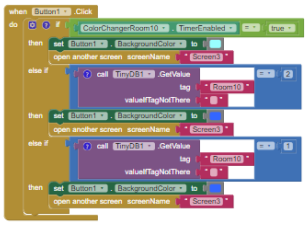


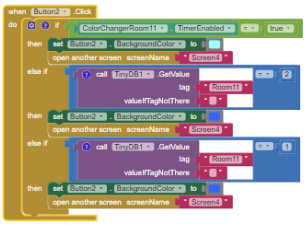

when Next .Click
do call TinyDB1 .StoreValue
tag button1
valueToStore get global but1
call TinyDB1 .StoreValue
tag button2
valueToStore get global but2
call TinyDB1 .StoreValue
tag button3
valueToStore get global but3
call TinyDB1 .StoreValue
tag button4
valueToStore get global but4
call TinyDB1 .StoreValue
tag button5
valueToStore get global but5
set Web1 . Url to join http:// call TinyDB1 .GetValue tag Room11IP valueIfTagNotThere 0
call Web1 .Get
set Web1 . Url to join http:// call TinyDB1 .GetValue tag Room10IP valueIfTagNotThere 0
call Web1 .Get
set global Room10 to 0
set global Room11 to 0
open another screen screenName textDisplay
when Room11IPaddressNotice .AfterTextInput
response
do set global room11IPadd to get response
call TinyDB1 .StoreValue
tag Room11IP
valueToStore join 192.168.137.
get global room11IPadd
when Room12 .Click
do if Room12 . BackgroundColor =
then set Room12 . BackgroundColor to
set global but3 to blue
else if Room12 . BackgroundColor =
then set Room12 . BackgroundColor to
set global but3 to red

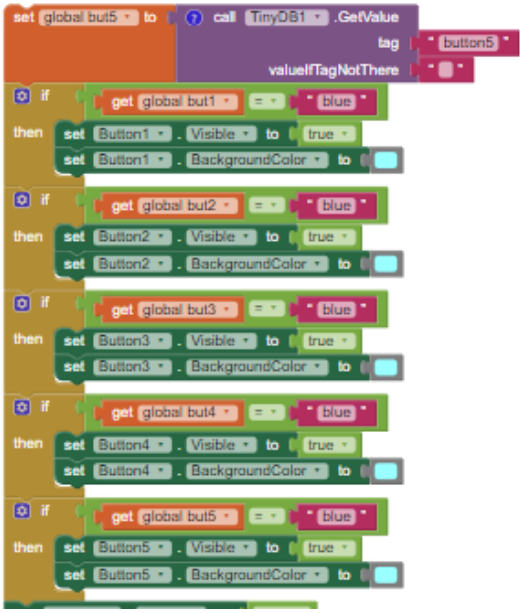

when Room11 .Click
do if Room11 . BackgroundColor =
then set Room11 . BackgroundColor to
set global but2 to blue
call Room11IPaddressNotice .ShowTextDialog
message Room 11's IP
title Please enter the last 2 or 3 numbers of 11's IP ....
cancelable true
else if Room11 . BackgroundColor =
then set Room11 . BackgroundColor to
set global but2 to red

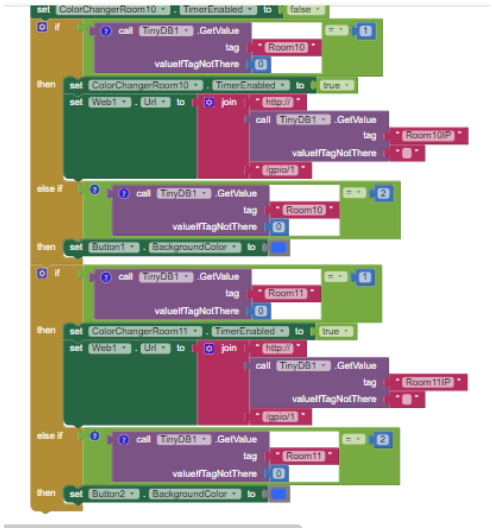

when Room13 .Click
do if Room13 . BackgroundColor =
then set Room13 . BackgroundColor to
set global but4 to blue
else if Room13 . BackgroundColor =
then set Room13 . BackgroundColor to
set global but4 to red

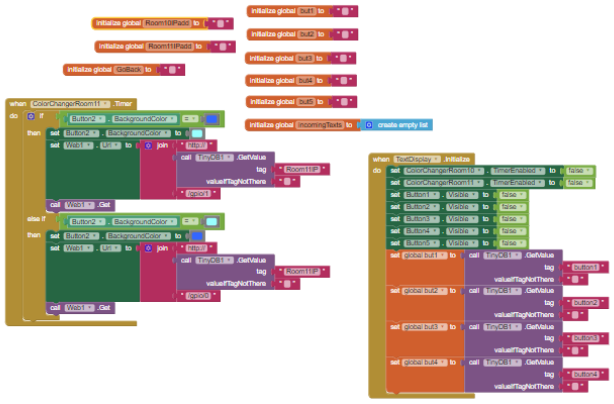
SCREEN 2


when Button1 .Click
do if ColorChangerRoom10 . TimerEnabled = true
then set Button1 . BackgroundColor to
open another screen screenName Screen3
else if call TinyDB1 .GetValue tag Room10 valueIfTagNotThere = 2
then set Button1 . BackgroundColor to
open another screen screenName Screen3
else if call TinyDB1 .GetValue tag Room10 valueIfTagNotThere = 1
then set Button1 . BackgroundColor to
open another screen screenName Screen3

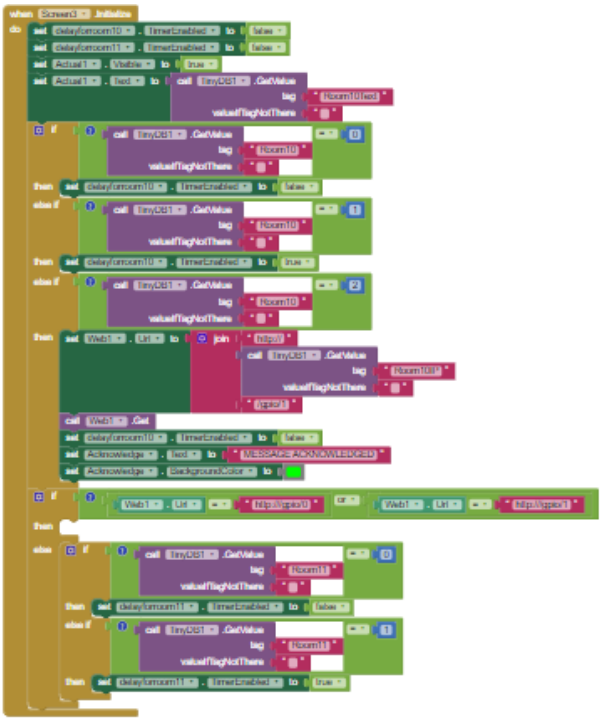

when Button2 .Click
do if ColorChangerRoom11 . TimerEnabled = true
then set Button2 . BackgroundColor to
open another screen screenName Screen4
else if call TinyDB1 .GetValue tag Room11 valueIfTagNotThere = 2
then set Button2 . BackgroundColor to
open another screen screenName Screen4
else if call TinyDB1 .GetValue tag Room11 valueIfTagNotThere = 1
then set Button2 . BackgroundColor to
open another screen screenName Screen4

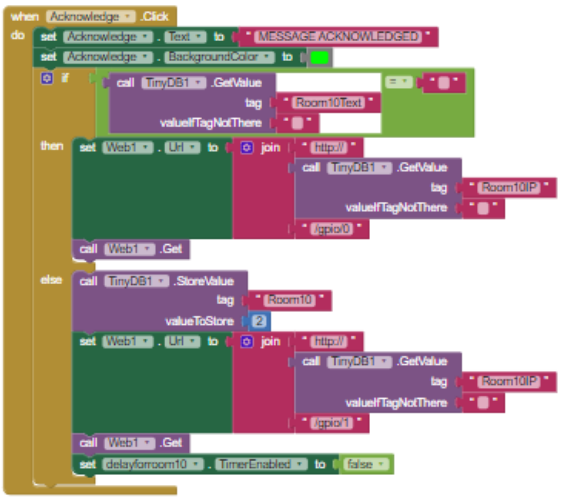

set global but5 to call TinyDB1 .GetValue
tag button5
valueIfTagNotThere
if get global but1 = blue
then set Button1 . Visible to true
set Button1 . BackgroundColor to
if get global but2 = blue
then set Button2 . Visible to true
set Button2 . BackgroundColor to
if get global but3 = blue
then set Button3 . Visible to true
set Button3 . BackgroundColor to
if get global but4 = blue
then set Button4 . Visible to true
set Button4 . BackgroundColor to
if get global but5 = blue
then set Button5 . Visible to true
set Button5 . BackgroundColor to

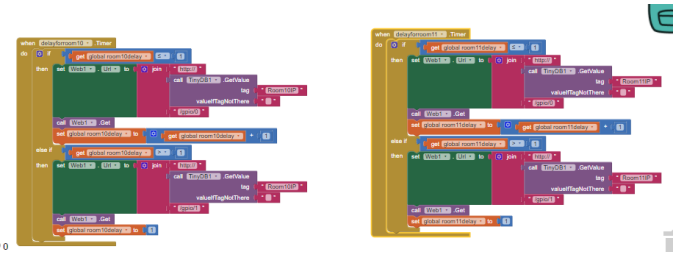

set ColorChangerRoom10 . TimerEnabled to false
if call TinyDB1 .GetValue tag Room10 valueIfTagNotThere 0 = 1
then set ColorChangerRoom10 . TimerEnabled to true
set Web1 . Url to join http:// call TinyDB1 .GetValue tag Room10IP valueIfTagNotThere /api/v1
else if call TinyDB1 .GetValue tag Room10 valueIfTagNotThere 0 = 2
then set Button1 . BackgroundColor to
if call TinyDB1 .GetValue tag Room11 valueIfTagNotThere 0 = 1
then set ColorChangerRoom11 . TimerEnabled to true
set Web1 . Url to join http:// call TinyDB1 .GetValue tag Room11IP valueIfTagNotThere /api/v1
else if call TinyDB1 .GetValue tag Room11 valueIfTagNotThere 0 = 2
then set Button2 . BackgroundColor to


initialize global Room10Pass to
initialize global Room11Pass to
initialize global GoBack to
initialize global but1 to
initialize global but2 to
initialize global but3 to
initialize global but4 to
initialize global but5 to
initialize global incomingText to create empty list
when ColorChangerRoom11 . Timer
when TextDisplay . Initialize

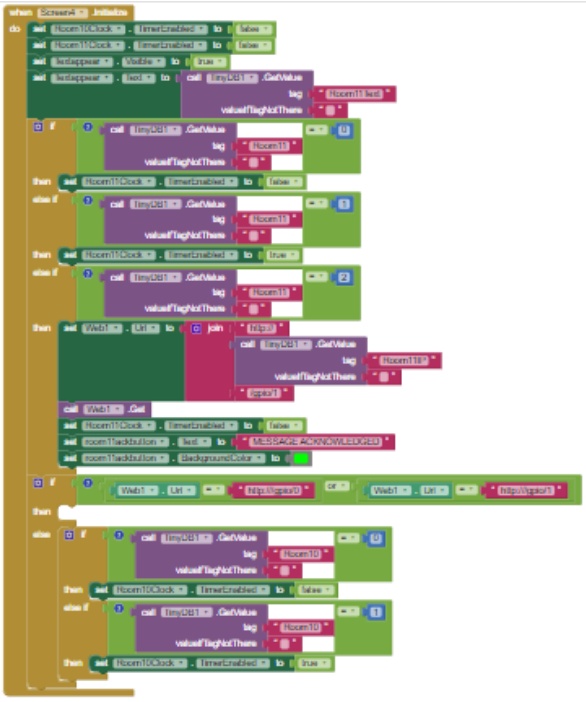
SCREEN 3

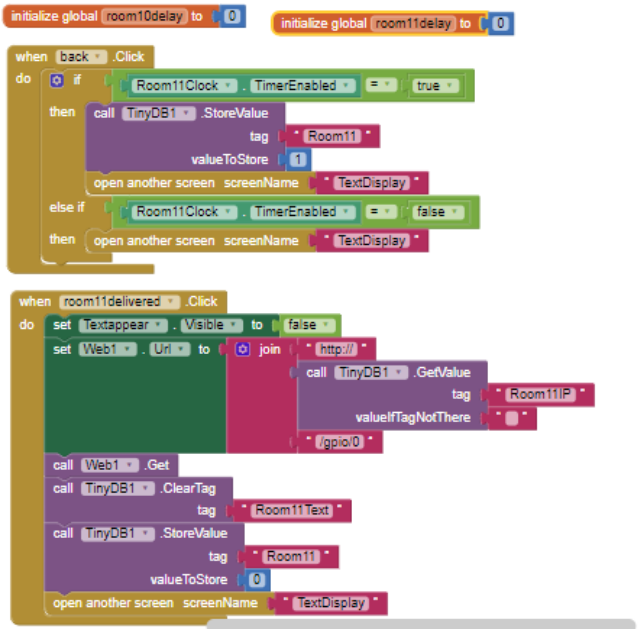


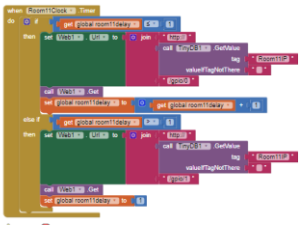


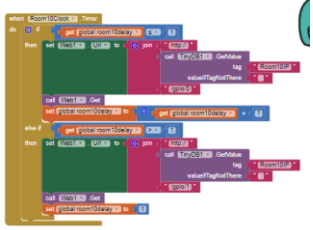


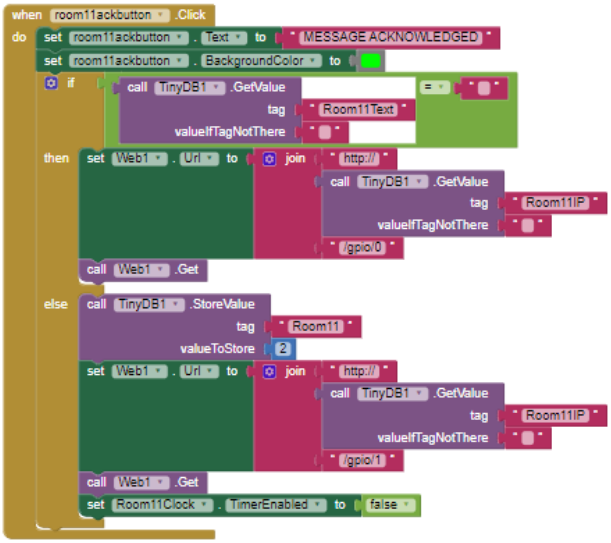
SCREEN 4




initialize global room10delay to 0
initialize global room11delay to 0
when back .Click
do if Room11Clock . TimerEnabled = true
then call TinyDB1 .StoreValue
tag Room11
valueToStore 1
open another screen screenName TextDisplay
else if Room11Clock . TimerEnabled = false
then open another screen screenName TextDisplay
when room11delivered .Click
do set Textappear . Visible to false
set Web1 . Url to join http://
call TinyDB1 .GetValue
tag Room11IP
valueIfTagNotThere
/gpio/0
call Web1 .Get
call TinyDB1 .ClearTag
tag Room11Text
call TinyDB1 .StoreValue
tag Room11
valueToStore 0
open another screen screenName TextDisplay






when room11ackbutton .Click
do set room11ackbutton . Text to MESSAGE ACKNOWLEDGED
set room11ackbutton . BackgroundColor to
if call TinyDB1 .GetValue tag Room11Text valueIfTagNotThere =
then set Web1 . Url to join http:// call TinyDB1 .GetValue tag Room11IP valueIfTagNotThere /gpio/0
call Web1 .Get
else call TinyDB1 .StoreValue tag Room11 valueToStore 2
set Web1 . Url to join http:// call TinyDB1 .GetValue tag Room11IP valueIfTagNotThere /gpio/1
call Web1 .Get
set Room11Clock . TimerEnabled to false