Tanta University
Computer and Automatic Control
Engineering Department

Baby Monitoring System for Smart Cradle Using IOT

Team Members

Abdelrahman Samy Shehab Eldien
Mohamed Abdelmonem Fayed
Mohamed Refaat Shaban
Taha Maher Shoair
Ahmed Mohamed Elsensar
Ahmed Mohamed Elgebaly

Supervised By
Dr. Mohamed Abdallah Attia

Graduation Project Book
Computer Engineering and Automatic Control

Tanta University
2019/2020

Abstract

Nowadays one of the most significant challenges that faces many families is baby care.
Parents cannot continuously observe or monitor their babies all the time. Baby monitors help in reassuring millions of parents that their children are safe.
The idea in This Project is to design a system that will simplify the process of monitoring the baby by using Raspberry Pi device. Thus, we have come up with an idea to design a Smart Cradle System using IOT which will help the Parents to monitor their child even if they are away from home & detect every activity of the Baby from any distant corner of the world.

The proposed system will have many features such as:
Measuring the Baby cradle temperature and humidity, displaying live video and audio, determine if the baby is awake or sleep, recording audio and music and playing it to the baby, and the ability to listen to the baby noise through the sound detection and take actions.

Keywords

Smart baby cradle, Raspberry PI, measuring temperature and humidity, live video and Audio, cry detection, and play on music and audio.

Contents Page

List of Figures Page

Chapter 1

1. Introduction To BMSCS

1.1 Introduction

The Baby Monitoring System for Smart Cradle Using IOT (BMSSC) allows parents to work without any worries about their The Baby and assist them to keep an eye on them remotely via application on mobile phone (android or iOS) system or via the smart baby cradle at home that contains LCD to display the humidity and the temperature changes.This system considers all the minute details required for the care and protection of the Baby in the cradle.

The design of smartness & innovation comes with the use of technologies/methodologies which include Internet of Things (IOT) (Modules like Raspberry Pi, Humidity & Temperature sensing), Swing Automation, Live Video Surveillance, firebase (Data Storage) & User-Friendly Android and iOS Mobile Application (for User Controls).

we have designed a baby monitoring system using Raspberry Pi whereas all the previous systems were developed using either Microcontroller or Arduino.

The Raspberry Pi is a full-fledged credit card sized computer which consists of 512 MB RAM and 700 MHz microprocessor while Arduino is 8-bit AVR microcontroller-based board which comprises of hardware prototype platform and Arduino language along with IDE and libraries. Raspberry Pi B+ module is used in this project which has a great advantage over microcontroller based projects. All the data which is been taken from the sensors/modules will be stored in Cloud (Google Firebase).

Even so there are various products that support baby monitoring, many of them are not efficient enough. Therefore, the aim of this project is to build a low-cost system that provides high-quality features.

1.2 The History Of Smart Baby Cradle

There are number of systems available for some kinds of cradle that help parents to monitor their baby. each system supports some features and missing other important ones. In the next subsections, we explain a number of these systems in terms of the functionalities and limitations

1. Integrity Baby Monitoring System [2014]

The design of a GSM-based smart baby monitor system aims to provide better care for children.

This system observes important parameters such as body temperature, humidity status, pulse rate, movement of an infant, and the use of GSM network. These Parameter details can be sent to the parents with alarm so any action can be taken. The system design consists of sensors for observing important factors, LCD display, GSM interface and sound alarm all controlled by one microcontroller[5].

This system has some disadvantages:

- **It cannot sense the room temperature and humidity.**
- **It does not have a mobile application**
- **It does not have live video and audio streaming.**

2. Advanced Baby Monitor [2017]

This project presents the design of advanced baby monitoring system using Raspberry Pi. Parameters such as humidity, temperature; movements of the infant are also monitored and the baby's sleep and sleep sequences are automatically recorded as a means by which parents could remotely observe and monitor their baby. A camera is used so that the observer and the parents can view it. This system architecture consists of sensors to monitor important parameters such as temperature and humidity sensor, motion sensor and sound sensor, which includes s microphone. The details of the parameter are sent to the parents by the alarm so if any action occurs can be taken [6].

This system has some disadvantages:

- **It does not have a mobile application**
- **It does not have Live audio streaming**

3. A Real-Time Infant Health Monitoring System for tough Hearing Parents [2017]

The system presented a real-time infant monitoring system for tough hearing parents by using mobile devices based on Android operating systems, which has sensors such as finger heartbeat, body temperature, humidity and sound detection [7].

In addition to, a microcontroller and android devices such as smartphone and smartwatch. In particular, this system is designed to monitor physiological information obtained from children and then produces alarms in case of abnormal situations. The implementation of this system depends on one of the Arduino boards which is the Leonardo board. This device is used to collect and sense information using the connected sensors and then create an appropriate alarm based on this data.

Smartwatches and smartphones based on Android operating system were used to report alarms to the parent. From the implementation results, the data collected were observed and sensed by the appropriate sensors which also contain any abnormal conditions and finally the alarms were notified.
This system has some disadvantages:

· The hardware connected to a mobile application via Bluetooth. This is not good due to the Poor Security, Battery Drain in the phone and short distance.

· It does not provide cry detection feature.

From this review, we have some different researches each support different features. Each system supports some features and missing other important ones. The target of this project is to propose a smart baby monitoring system that overcome the disadvantages existing on these systems. In order to gather as many important features as possible in one system.

1.2.1 Project Scope

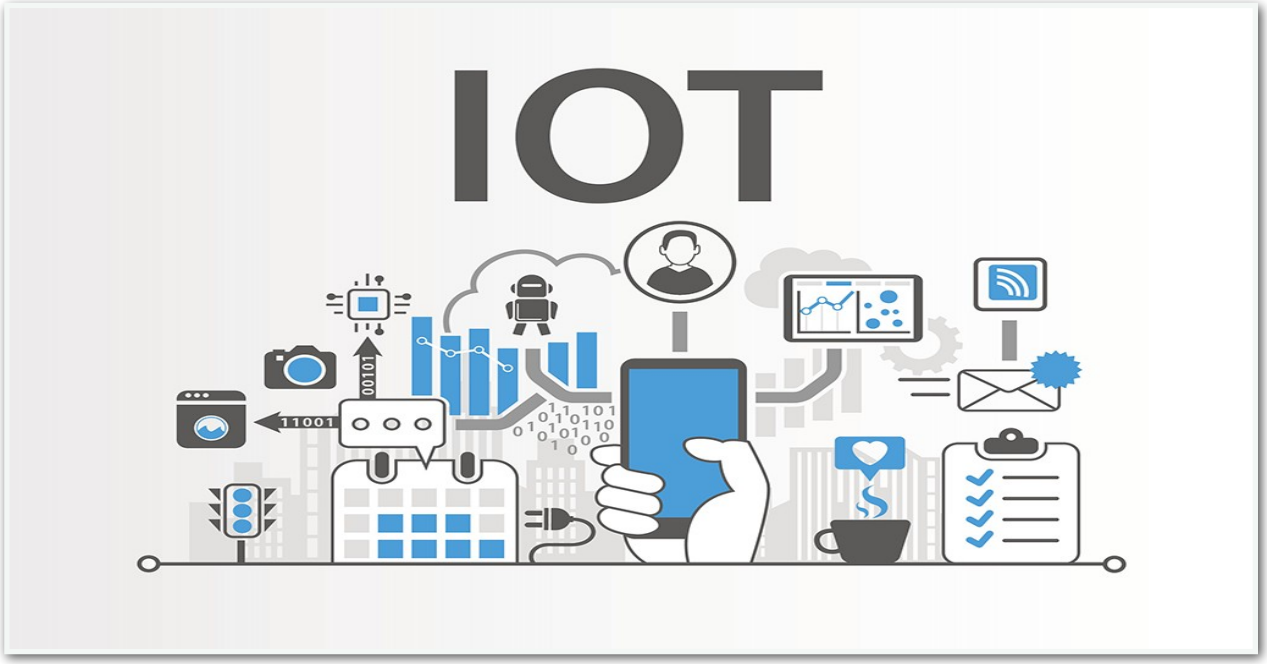
The proposed smart baby cradle IOT. This work aims to produce a baby monitoring system, which provides the following important features:

- **Live video and audio streaming of the baby.**
- **Room temperature and humidity observation.**
- **Play music for baby.**
- **Notifications when temperature goes to high.**
- **Crying detection if the sound is high**
- **Use Mobile Application to communicate with parents.**
- **Taking Actions to the baby using application connected to server.**

Chapter 2

2. IOT System Structure

2.1 Introduction to IOT



The phrase Internet of Things (IoT) refers to connecting various physical devices and objects throughout the world via internet. The term IoT was firstly proposed by Kevin Ashton in 1999.The following section illustrates IoT basics [2].

It deals with various layers used in IoT and some basic terms related to it. It is basically expansion of services provided by Internet.

This section also presents the architecture of IoT. For example, when the household devices of our daily life connect with the internet the system can be called a Smart-Home in IoT environment.

The IoT is not just deep vision for future. It is already under implementation and is having an impact on more than just technological development. It provides different devices the ability to transmit and receive data on the internet simply. One of the most substantial roles for IoT is real-time monitoring so, the objective of this work is to create a smart baby monitoring system based on one of the powerful IoT devices, which is Raspberry Pi microcontroller.

In particular, it will focus monitoring the baby using camera, microphone, speaker, and sensor for measuring the surrounding temperature and humidity.

A critical requirement of an IoT is that the things in the network must be connected to each other. IoT system architecture must guarantee the operations of IoT, which connects the physical and the virtual worlds. Design of IoT architecture involves many factors such as networking, communication, processes etc[3].

The challenge will be to design a system that facilitates and provide high-quality monitoring process.

2.2 IOT Control Devices

Based on the availability of usable I/O interfaces for sensors, interfaces for Internet connectivity, memory and storage interfaces, and audio/video interfaces the following boards are evaluated. IoT devices can also be used to a variety of other purposes, for instance, wearable sensors, smart watches, LED lights, automobiles and industrial machines.

1. Arduino

Arduino is an 8-bit microcontroller development board with a USB programming interface to connect to a computer and additional connection sockets to external electronics like sensors, motors speakers, diodes etc. It has got both input and output pins, the input pins can be either be digital (0 – 13) or analogue (A0 – A5), while the output pins are only digital (0 – 13).

Arduino board design is open source and it also has an open source integrated development environment which has a cross-compiler, a debugger and a serial monitor to control the inputs and outputs. Arduino can either be powered through the USB connection from the computer, from a 9V battery, or from a power supply.

2. Raspberry Pi

Raspberry Pi is a computer-based development board which runs on a Linux distribution referred to as Raspbian Linux. It can work and be connected like any computer to a mouse, keyboard, and screen perform computing functions. Raspberry Pi B+ board has 32-bit processor, four USB ports, HDMI port, Ethernet port, Audio port, CSI camera connector, and micro SD card slot.

It also has 40 general-purpose input/output. Raspberry Pi comes in different models, Model 2 lacks an embedded Wi-Fi but a Wi-Fi adapter can be used via the USB port to get internet connectivity. New models of Raspberry Pi 3 have integrated Wi-Fi module in its board making it easier to configure internet connectivity.

3. ESP8266 - Node MCU

This is an open source development board with a firmware that runs on ESP8266 module. The ESP-8266 module is a wireless programmable microcontroller board. The ESP8266 Wi-Fi board is a SOC with integrated TCP/IP protocol stack that can give any secondary microcontroller access to a Wi-Fi network [5]. The ESP8266 board is capable of either hosting an application or offloading
all Wi-Fi networking functions from another application processor and therefore this is more suitable to be used as a sensing node that is capable to sense the data from various wirelessly connected IoT sensor nodes and send data to the central server like. In figure 2.1

	Arduino	**Raspberry Pi**	**ESP8266 Node MCU**
Developer	Arduino	Raspberry Pi Foundation	ESP8266 open source community
Type	Single board microcontroller	Mini computer	Single board microcontroller
Operating System	None	Linux	XTOS
CPU	Atmel, ARM, Intel	ARM Cortex	LXT106
Clock Speed	16 MHz	1.2GHz	26 MHz – 52 MHz
Memory	32KB	1-4GB	Upto 128MB
Storage	1KB	MicroSDHC Slot	4MB
Power	USB, Battery, Power Supply	USB, Power Supply	USB
Operating Voltage	5V	5V	3.3V
I/O Connectivity	SPI I2C UART GPIO	SPI DSI UART SDIOCSI GPIO	UART, GPIO

Table 2.1: Comparison of Arduino, Raspberry Pi and ESP8266 Boards

2.3 Web services

The term "Web service" describes a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet Protocol backbone. XML is the data format used to contain the data and provide metadata around it, SOAP is used to transfer the data, WSDL is used for describing the services available and UDDI lists what services are available.

A Web service is a method of communication between two electronic devices over a network. It is a software function provided at a network address over the Web with the service always-on as in the concept of utility computing.

Many organizations use multiple software systems for management Different software systems often need to exchange data with each other, and a Web service is a method of communication that allows two software systems to exchange this data over the Internet. The software system that requests data is called a service requester, whereas the software system that would process the request and provide the data is called a service provider.

Different software may use different programming languages, and hence there is a need for a method of data exchange that doesn't depend upon a particular programming language. Most types of software can, however, interpret XML tags. Thus, Web services can use XML files for data exchange.

2.3.1 Web services (generic)

Asynchronous JavaScript And XML (AJAX) is a dominant technology for Web services. Developing from the combination of HTTP servers, JavaScript clients and Plain Old XML (as distinct from SOAP and W3C Web Services), now it is frequently used with JSON as well as, or instead of, XML.

REST:

Representational State Transfer (REST) is an architecture for well-behaved Web services that can function at Internet scale.

In a 2004 document, the W3C sets following REST as a key distinguishing feature of Web services:

We can identify two major classes of Web services:

- REST-compliant Web services, in which the primary purpose of the service is to manipulate XML representations of Web resources using a uniform set of stateless operations; and
- arbitrary Web services, in which the service may expose an arbitrary set of operations.

Web services that use markup languages

There are a number of Web services that use markup languages:

- **JSON-RPC.**
- **JSON-WSP**
- **Representational state transfer (REST) versus remote procedure call (RPC)**
- **Web Services Conversation Language (WSCL)**
- **Web Services Description Language (WSDL), developed by the W3C**
- **Web Services Flow Language (WSFL), superseded by BPEL**
- **Web template**
- **WS-Metadata Exchange**
- **XML Interface for Network Services (XINS), provides a POX-style web service specification format**

2.3.2 Web API

A Web API is a development in Web services where emphasis has been moving to simpler representational state transfer (REST) based communications. Restful APIs do not require XML-based Web service protocols (SOAP and WSDL) to support their interfaces.

2.3.3 W3C Web services

In relation to W3C Web services, the W3C defined a Web service as: A web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards.W3C Web Services may use SOAP over HTTP protocol, allowing less costly (more efficient) interactions over the Internet than via proprietary solutions like EDI/B2B. Besides SOAP over HTTP, Web services can also be implemented on other reliable transport mechanisms like FTP. In a 2002 document, the Web Services Architecture Working Group defined a Web services architecture, requiring a standardized implementation of a "Web service".

2.4. Applications

2.4.1 Swift

It is programming language that has potential to reshape in the future.It was released by Apple Inc. in June 2014 for iOS (and supporting systems) and Linux. It is the primary programming language used for developing iOS and OS X apps.

Swift is the fastest growing language, according to TNW. The demand for Swift developers has increased 600 percent making them the most hired developers.

Google is also considering to make Swift its first-class language instead of Java. If Google shifts to Swift, the demand for Swift apps and developers will skyrocket and there will be no other competing language.

For now, Swift is only available for iOS development but since it works on Linux and is open source, which means it can be used by anyone. It is still new and those who will shift to Swift early will have the advantage.

Key features:

- **Extremely easy to learn especially if you know Objective-C.**
- **It is open source.**
- **It is a simplified version of Objective-C.**
- **Easy-to-code.**
- **Maintenance is super-easy.**
- **It is the future of iOS development.**
- **It needs less coding as compared to other languages**
- **Modern**
- **Designed for safety**
- **Fast and Powerful**

1. Swift is faster :

The performance of Swift is almost the same as that of C++, which is considered the fastest in algorithm calculation arithmetic. Apple had this idea in mind and worked to improve the speed of Swift. For example, Swift 2.0 has beaten C++ in several computation algorithms, such as the Mandelbrot algorithm

2.Swift is safer :

Nowadays, an app's data security is a substantial characteristic of a successful product. The construction of Swift has been designed to exclude and avoid mistakes with the help of its features – generics, optional, and type interference to achieve app stability. Therefore, apps developed in Swift are less prone to bugs and crashes.

3.Swift is more readable :

To start, the code in Swift more closely resembles English, making it easier to read and requiring less time to check the code. As well, in

general, it requires far fewer lines of code for the same feature. Swift is easy to read by JavaScript, Java, Python, C#, and C++ programmers who are able to use it to some extent.

4. Swift has less code :
Swift is a more compact language for programming. However, this fact doesn't imply code simplicity, of course. At times, it can be very difficult to write, but it brings more benefits and is highly reusable.

5. Swift is closer to other platforms:
This point is very important, especially when speaking about the cooperation between programmers building the same app on different platforms.

6. Swift is Apple's ongoing focus:
Apple Inc. is concentrating on evolving Swift as its core programming language. Recently, the WWDC (Worldwide Developers Conference) gave a detailed presentation on Swift 4.2 where they presented a number of great features and updates. The current Swift version has:

- **Faster builds**
- **Language features to improve efficiency and remove boilerplate SDK improvements**
- **Converging towards binary compatibility**

Moreover, Swift is expected to complete an important milestone in 2019 -binary compatibility with future Swift compiler releases.

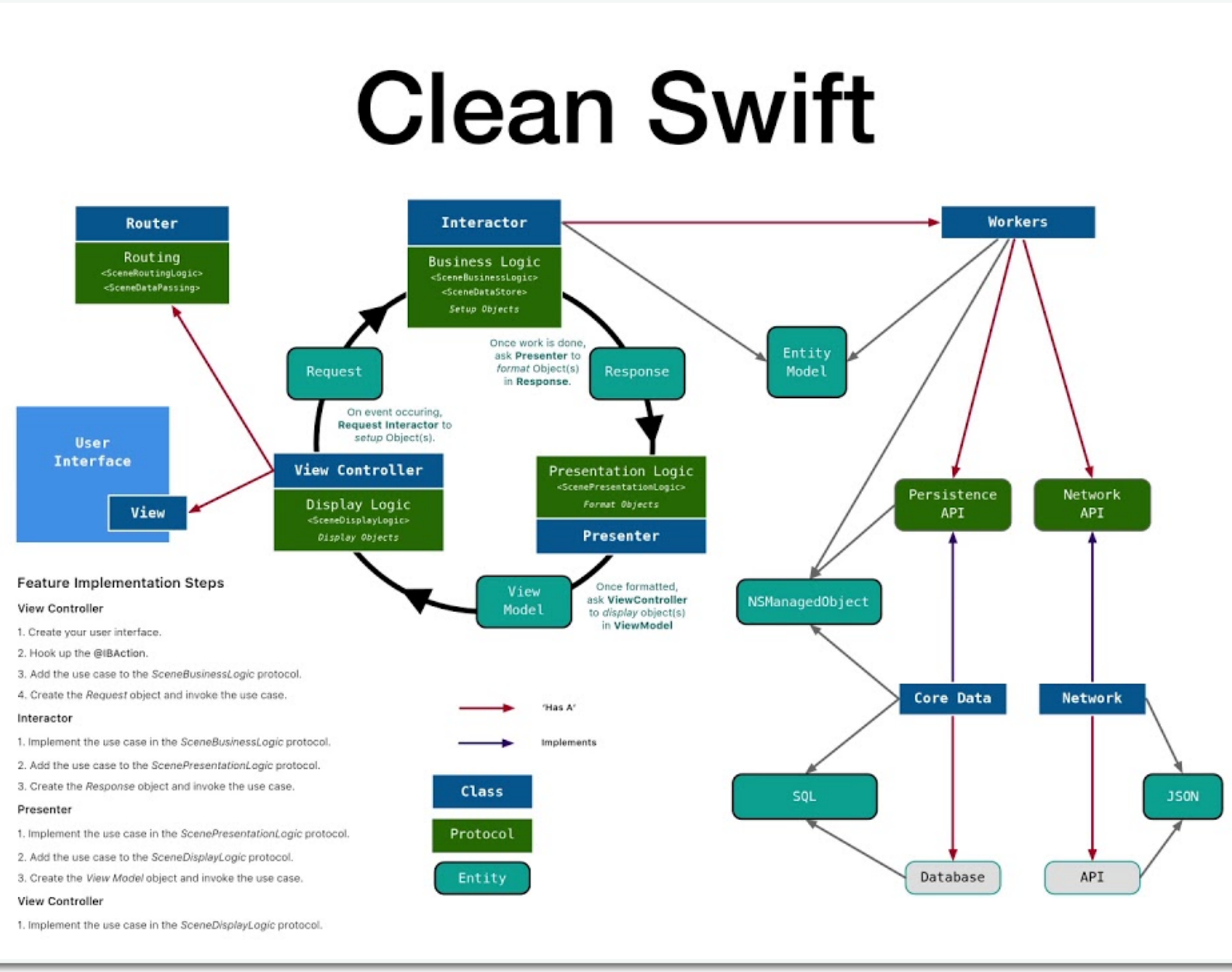
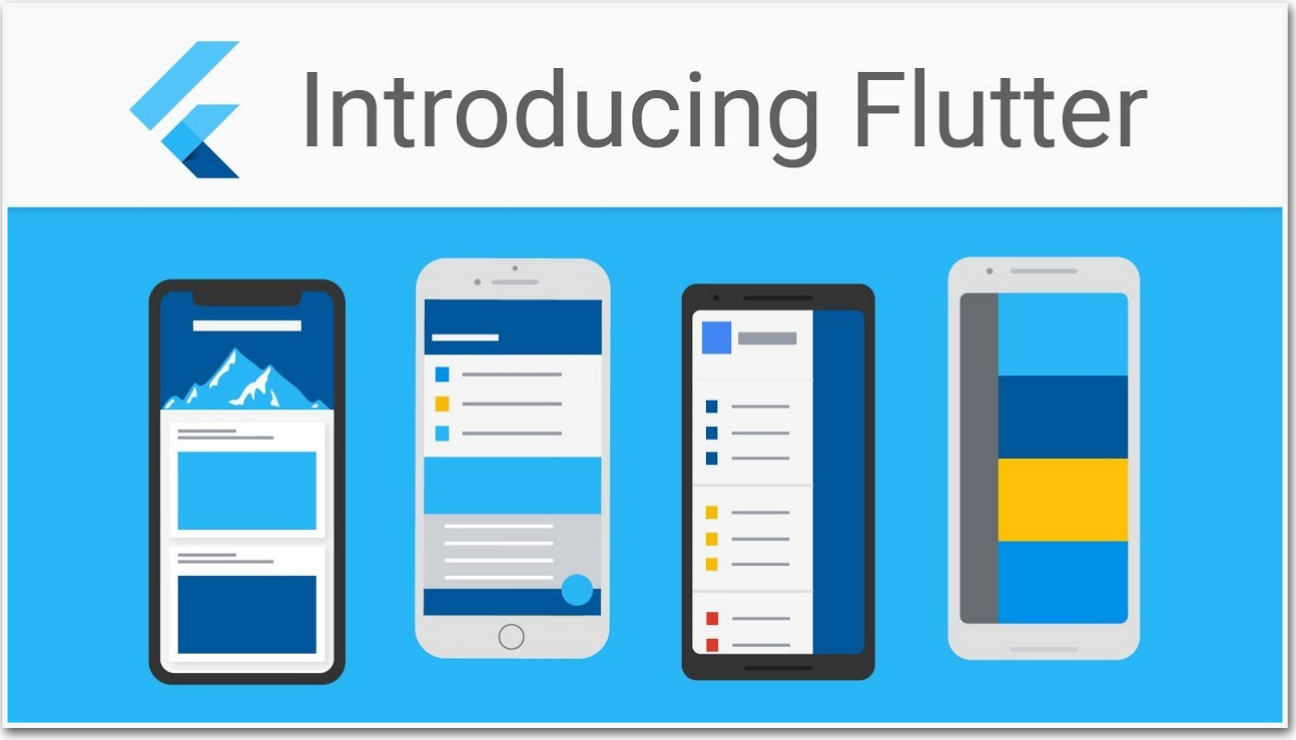


Figure 2.2: Clean Swift Structures

2.4.2 Flutter



Packed of hearing "Flutter" everywhere for the app development? Being in a technical environment, we all have heard enough about Google launched an open-source cross-platform framework & software development kit also known as Flutter SDK which is quite in trend nowadays. There are lots of cross-platform frameworks for ages, then why developers started using a newbie cross-platform framework that has barely completed a couple of years?

Well, do you know why Flutter is so trending and has become a word of the mouth of everyone?

You may have heard about Flutter, but never have understood its characteristics for what it is on the top today. So, let's get to know what Flutter is exactly and the so-called properties that make it lead everywhere.

Flutter Progress So Far

Flutter is a free and open-source mobile UI framework created by Google. In a few words, it allows you to create a native mobile application with only one codebase. This means that you can use one programming language and one codebase to create two different apps (for iOS and Android).

Flutter consists of two important parts:

1) An SDK (Software Development Kit): A collection of tools that are going to help you develop your applications. This includes tools to compile your code into native machine code (code for iOS and Android).
2) A Framework (UI Library based on widgets): A collection of reusable UI elements (buttons, text inputs, sliders, and so on) that you can personalize for your own needs.

Flutter was released on the o4 December 2018, at the Flutter Live event denoting the very first "stable" version of the Google present cross-platform framework. The framework has kept coming with new versions of its including more features and innovative approaches every time.

Flutter is continuously appearing out as an ideal cross-platform framework with more advanced features since its stable version. Talking about its updates & modifications, lets what significant growth and updates have seen so far.

The latest version Flutter 1.12, released in the Keynote interaction in the last month of 2019, was the craziest moment for flutter lovers as this version is, going to involve superior features and enhancements. Have a glance at some of them below:

1) iOS Dark mode unable
2) AndroidX
3) Add-to-App Support
4) Google fonts support
5) Supernova free for Flutter
6) Adobe XD included
7) Web Support on beta
8) MacOS Support Alpha version
9) Dart 2.7 roll out with more improvements

Key Factors of Flutter:

Well, Flutter's new versions will keep coming with more advanced and variety of features of the time. But many other factors are

there in-built in Flutter since its first release those are unforgettable, which have played important roles to take Flutter to the next level and have made it so popular for the cross-platform mobile app and progressive web app development.

We have curated and mentioned-below those key factors which are standing behind the Flutter's fam:

1) Single Code & Less Testing

When you have an audience on both the platforms iOS & Android that you can't sacrifice any of platform to develop your app, but the budget is limited, this is where Flutter arrives to rescue. Its magnificent quality of writing code once and use that to develop apps for multiple platforms with assured quality that even reduce the time to quality analysis and testing all in under your budget.

2) Fast Coding with Hot Reload

Besides, Flutter single code can be used in app development for multiple platforms, it also raises the coding speed for developers and makes the code writing process efficient & seamless altogether all because of its unbeatable Hot Reload feature. Hot Reload is as simple as modifying any code and seeing its result right away that lets developers change the codes multiple times whenever it’s required with the quick result and not to restart all the process again.

3) Custom, Animated Ui Compatible All Device Types Designs

The software development kit- Flutter lets you create customize and complex User Interfaces at once for all devices. Your flutter app will appear the same on every platform, be it iOS or Android and their

latest or old versions of devices, Flutter is capable of running smoothly, and it cuts off the need to add additional cost to support the various devices.

4) Single Code For Multiple OSs With Sustaining Native Experience

Many cross-platform may let you write the code once and employ it on both iOS and Android platforms, but not all can render the same look as a native app. As Flutter is the cross-platform platform that sustains the native experience and feel of the app, you don't have to worry about the app performance for any platform. The Flutter apps write in the Dart language that is capable of eliminating the JavaScript bridge and directly compiles to native machine code and launch the app faster than React Native.

5) Perfect For MVP

Minimum Viable Product is the ideal way to validate the online business idea with the essential features and offerings. The MVP needs to be released at the initial stage and before you develop and deploy the full-fledged app, here, Flutter works effectively. Flutter's characteristics such as high speed, quick and easy integration, and customizable UI, lower cost & time consumption have made it the perfect choice for the MVP. To launch a new app or for an online startup business, entrepreneurs tend to hire Flutter developers to make their MVP convincing and cost-effective.

❖ Native Apps

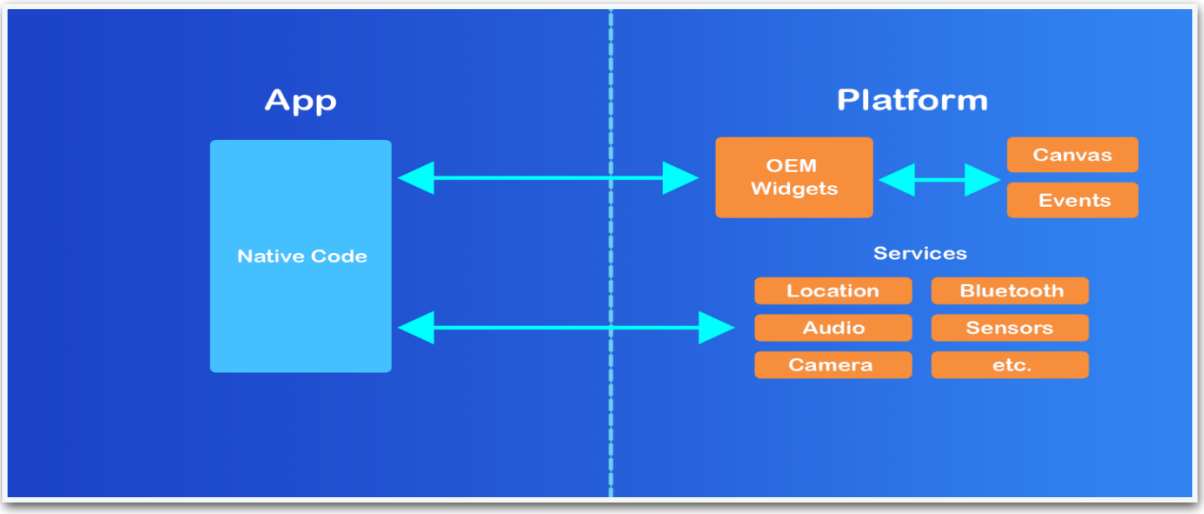


Figure 2.3: **Native Apps Architecture**

In the native apps, whether it is Android or iOS, the native app code talks to the Platform to create OEM Widgets or for accessing various Services like audio, sensors, camera, etc. The widgets are rendered to a screen canvas, and events are passed back to the widgets. This architecture restricts the widgets to be reused across all platforms as they are OEM specific. And, this is the reason we have to write whole app for each platform separately.

❖Flutter App

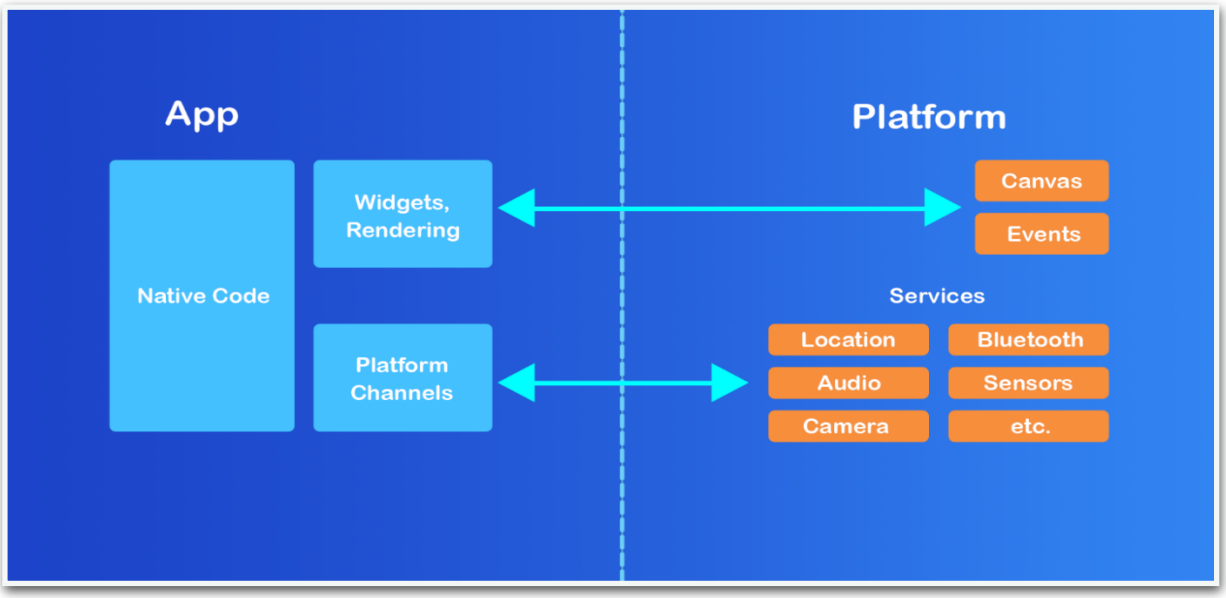


Figure 2.4: **Flutter Architecture**

Now, coming to the Flutter apps. Flutter solves the most challenging part of the other cross-platform frameworks, i.e. getting rid of the BRIDGE. Flutter does not use the OEM widgets; it provides its own widgets. Flutter moves the widgets and the renderer from the Platform into the app, which allows them to be customizable and extensible. This also helps Flutter to maintain the consistent 60 FPS.

Overview of Flutter vs Native

The biggest advantage that Flutter gives in comparison to native Android is the cross-platform support, i.e., you can use the same codebase for different platforms like Android, iOS, Web, Desktop, etc.

But can it deliver the same performance and stability like native app does? When it comes to cross-platform this is the most important question that all developers have in their mind. So, let me say that cross-platform apps still might not be as stable as native apps nor it can give the same level of performance like native apps in certain situations.
Now, the question comes, how can a cross-platform app be as performant as a native app? This is due to the excellent architecture of Flutter Framework and due to the language used by Flutter, i.e., Dart.

➢Dart:

Another reason that developers love Flutter is because of the Dart Language.
Dart is an object-oriented, class defined, garbage-collected language using a C-style syntax that trans-compiles optionally into JavaScript.
Some of the features that separate Dart from the other languages are as follows:

1) Dart uses AOT (Ahead of Time) compilation, which gives the fast startup and fully customizable Flutter widgets.

2) Dart also uses JIT (Just in Time) compilation, which is the main reason that Hot Reload exists. I will talk about it in a bit.

3) Dart has a garbage collector built in the language. This enables Flutter to achieve smooth animations and transitions that run at 60fps.

4) Dart allows Flutter to avoid the need of a separate declarative layout language like XML in Android, or separate visual interface builders, because Dart's declarative, programmatic layout is easy to read and visualize.

5) Dart has an easy learning curve, because it has similarity with various other languages. It is the combination of the features of these languages that makes Dart so powerful.

Can Flutter dominate over native?

Put it all together, Flutter is the most potential cross-platform framework and SDK that is the reason behind its rising popularity. It is much easier to learn than React Native, besides, the most eligible features hot reload, less coding, customize widgets, and many more are making it favorable to developers. As it writes in the object-oriented programming language Dart, but it still not production ready for certain applications.
It lacks in availability of certain plugins. Some of the important plugins that are available, are still buggy and are not usable in large scale production application. But this highly depends on the kind of app you or your company would be working on, if you get good plugins and it satisfies your app's feature needs, then Flutter might be the best choice for you.

2.4.3 Angular

Angular is a platform and framework for building single-page client applications using HTML and TypeScript.
Angular is written in TypeScript. It implements core and optional functionality as a set of TypeScript libraries that you import into your apps.

architecture of an Angular application relies on certain fundamental concepts. The basic building blocks are NgModules, which provide a compilation context for components. NgModules collect related code into functional sets; an Angular app is defined by a set of NgModules.

An app always has at least a root module that enables bootstrapping, and typically has many more feature modules.

- **Components define views, which are sets of screen elements that Angular can choose among and modify according to your program logic and data.**
- **Components use services, which provide specific functionality not directly related to views.**

Both components and services are simply classes, with decorators that mark their type and provide metadata that tells Angular how to use them.

- **The metadata for a component class associates it with a template that defines a view. A template combines ordinary HTML with Angular directives and binding markup that allow Angular to modify the HTML before rendering it for display.**
- **The metadata for a service class provides the information Angular needs to make it available to components through dependency injection (DI).**

An app's components typically define many views, arranged hierarchically. Angular provides the Router service to help you define navigation paths among views. The router provides sophisticated in-browser navigational capabilities.

Modules

Angular NgModules differ from and complement JavaScript (ES2015) modules. An NgModule declares a compilation context for a set of components that is dedicated to an application domain, a workflow, or a closely related set of capabilities. An NgModule can associate its components with related code, such as services, to form functional units.

Every Angular app has a root module, conventionally named AppModule, which provides the bootstrap mechanism that launches the application. An app typically contains many functional modules.

Like JavaScript modules, NgModules can import functionality from other NgModules, and allow their own functionality to be

exported and used by other NgModules. For example, to use the router service in your app, you import the Router NgModule.

Organizing your code into distinct functional modules helps in managing development of complex applications, and in designing for reusability. In addition, this technique lets you take advantage of lazy-loading—that is, loading modules on demand—to minimize the amount of code that needs to be loaded

---

Components

Every Angular application has at least one component, the root component that connects a component hierarchy with the page document object model (DOM). Each component defines a class that contains application data and logic, and is associated with an HTML template that defines a view to be displayed in a target environment.
The @Component() decorator identifies the class immediately below it as a component, and provides the template and related component-specific metadata.

Routing

The Angular Router NgModule provides a service that lets you define a navigation path among the different application states and view hierarchies in your app. It is modeled on the familiar browser navigation conventions:

- **Enter a URL in the address bar and the browser navigates to a corresponding page.**
- **Click links on the page and the browser navigates to a new page.**
- **Click the browser's back and forward buttons and the browser navigates backward and forward through the history of pages you've seen.**

The router maps URL-like paths to views instead of pages. When a user performs an action, such as clicking a link, that would load a new page in the browser, the router intercepts the browser's behavior, and shows or hides view hierarchies.

If the router determines that the current application state requires particular functionality, and the module that defines it hasn't been loaded, the router can lazy-load the module on demand.

To define navigation rules, you associate navigation paths with your components. A path uses a URL-like syntax that integrates your program data, in much the same way that template syntax integrates your views with your program data. You can then apply program logic to choose which views to show or to hide, in response to user input and your own access rules.

Chapter 3

3. Design and Fabrication of Smart Cradle

3.1 Hardware Components

The information regarding the components required in the baby cradle was decided to ensure that they can be installed without errors. We also surveyed available baby cradles that included a baby monitoring system in the market to gain some
insights into the structure of the baby cradle. During this phase, the hardware and software components used in this study were selected.

The hardware components included the following:

1. Raspberry Pi 3 Model B+
2. Raspberry Camera Module v2
3. DHT11-Temperature and Humidity Sensor
4. USB Microphone
5. 16 GB Micro SD Card
6. Speaker
7-power supply 12V 1.25A
8-Fan 10*10cm 12Vdc
9-LCD 2*16
10-Sound Detection Sensor
11-DC Motor 12Vdc
12-Kit L298 Board
7. Premium jumper wires
9. Breadboard

Raspberry Pi 3 Model B+ :

The main concept is based on Atmel ATmega644 which is particularly designed for educational use and intended for Python[1].

A Raspberry Pi is of small size i.e., of a credit-card-sized single-board computer, which is developed in the United Kingdom (U.K) by a foundation called Raspberry Pi. The first generation of Raspberry (Pi 1) was released in the year 2012, which has two types of models namely model A and model B.

In the subsequent year, A+ and B+ models were released. Again in 2015, Raspberry Pi2 model B was released and an immediate year Raspberry Pi3 model B was released and we that Raspberry pi3 B+ in our project.

Raspberry Pi can be plugged into a TV, computer monitor, and it uses a standard keyboard and mouse. It is user-friendly as it can be handled by all the age groups.

It does everything you would expect a desktop computer to do like word-processing, browsing the internet spreadsheets, playing games to playing high definition videos.***In Figure 3.1***

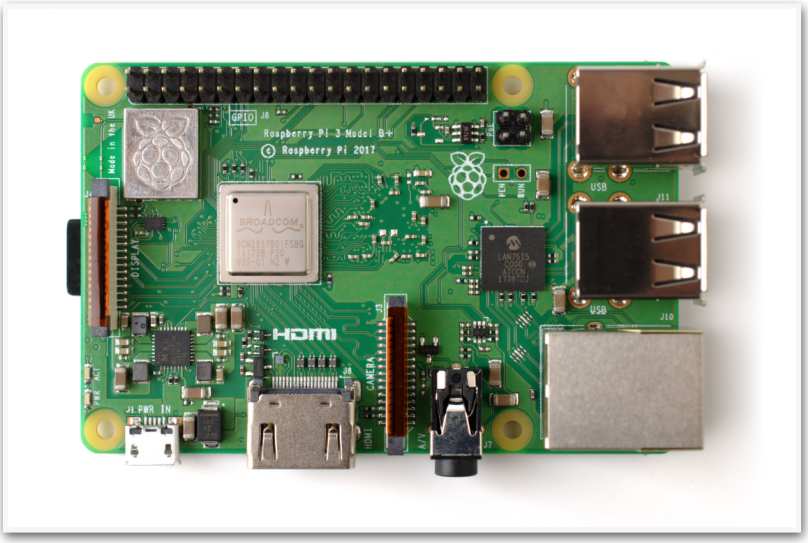


Figure 3.1: **Raspberry Pi 3 Model B+**

Raspberry Pi 3 Model B was released in February 2016 with a 1.2 GHz 64-bit quad core processor, on-board 802.11n Wi-Fi, Bluetooth and USB boot capabilities. On Pi Day 2018,
The **Raspberry Pi 3 Model B+** was launched with a faster 1.4 GHz processor and a three-times faster gigabit_Ethernet (throughput limited to ca. 300 Mbit/s by the internal USB 2.0 connection) or 2.4 / 5 GHz dual-band 802.11ac Wi-Fi (100 Mbit/s).

Other features are Power over Ethernet (PoE) (with the add-on PoE HAT), USB boot and network boot (an SD card is no longer required) and combined 3.5mm audio out jack and composite video and memory 1GB LPDDR2-900 SDRAM and have CSI camera port for connecting a Raspberry Pi camera. ***In Figure 3.2***

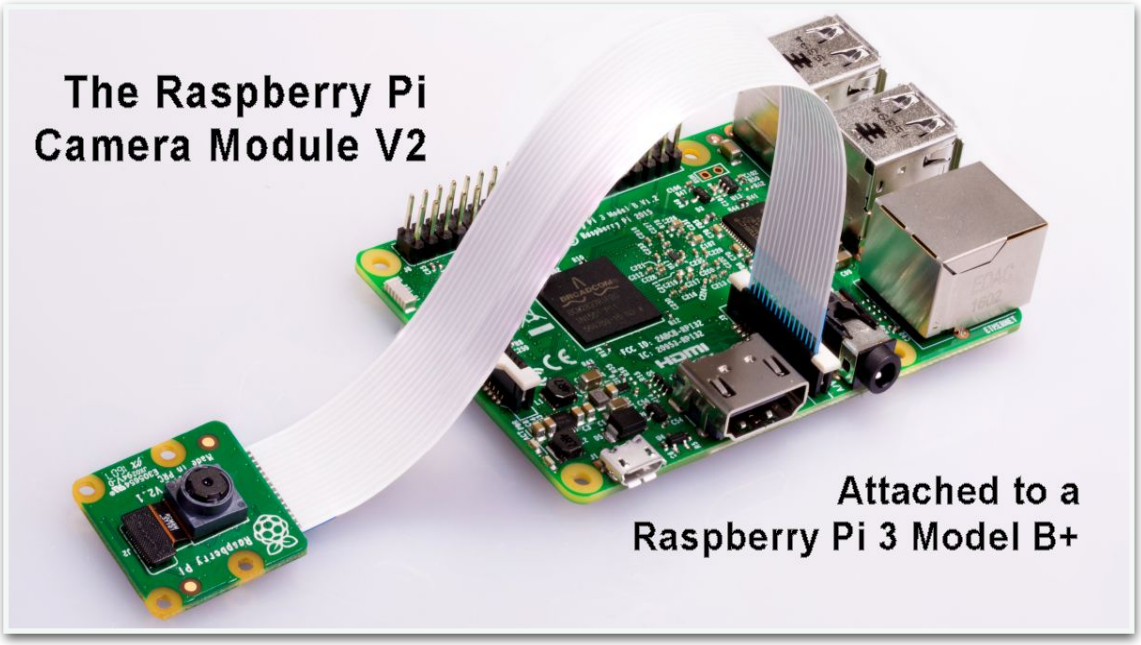


Figure 3.2: **Raspberry Camera Module v2**

DHT11-Temp& Hum Sensor:

DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output. By using the exclusive digital-signal-acquisition technique and temperature & humidity sensing technology, it ensures high reliability and excellent long-term stability.

This sensor includes a resistive-type humidity measurement component and an NTC temperature measurement component, and connects to a high performance 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability and cost-effectiveness. ***In figure 3.3***

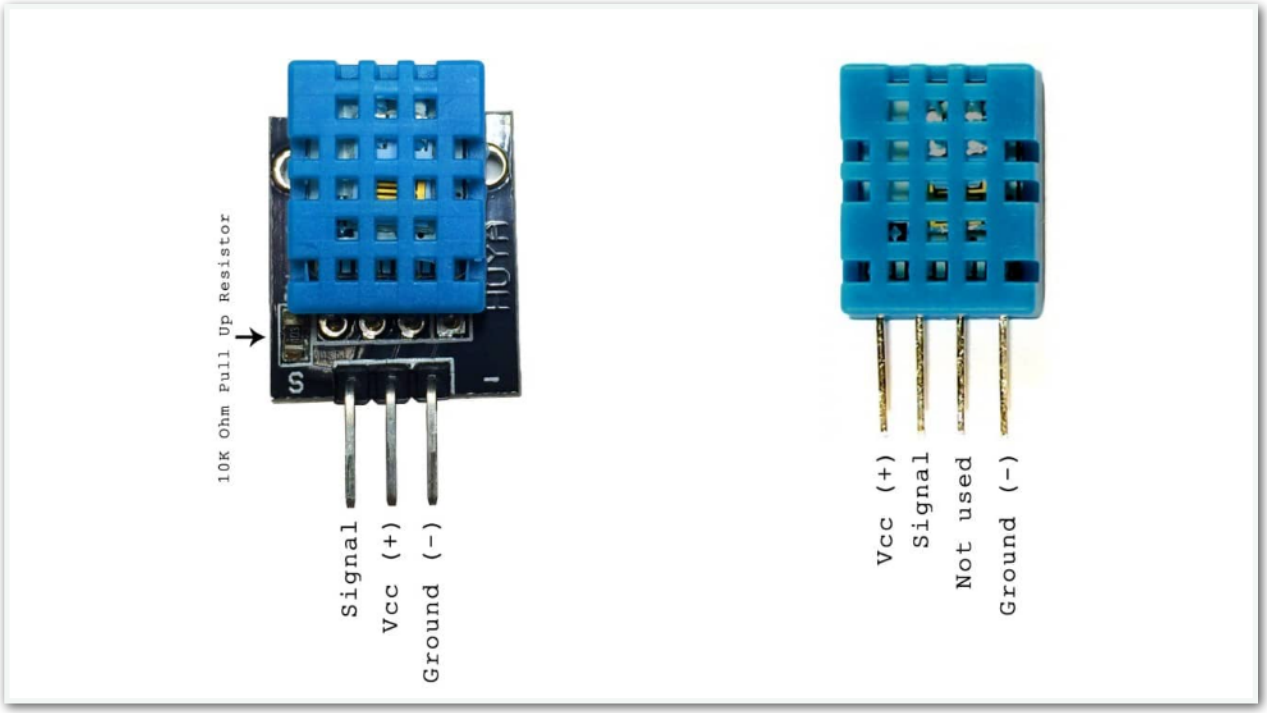


Figure 3.3: DHT11-Temperature and Humidity sensor

- **Sound Detection Sensor**

Features:

- Main chip: LM393, with polar microphone
- Working voltage: DC 4~6 V
- Signal output indicator.
- Single way signal output.
- Output signal is low level.
- When there is a sound, output low level, signal light is on.
- Can be used for voice activated lights, with light sensors to make a sound and light alarm, and in the place for voice control, voice detection. ***In figure 3.4***

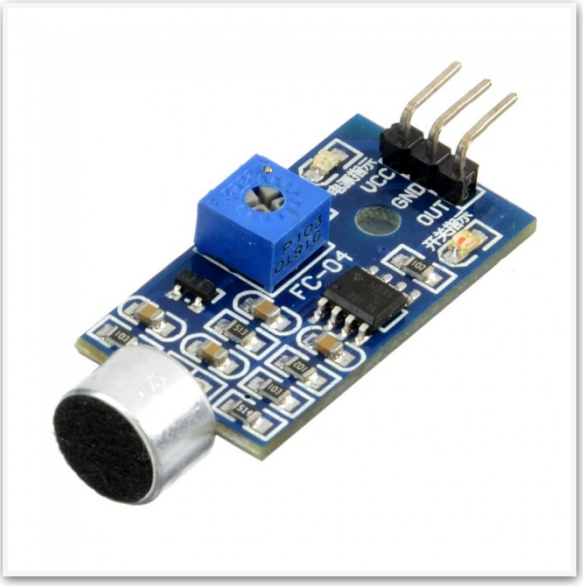


Figure 3.4: **Sound Detection Sensor**

- **DC Geared Motor.**

Specification of geared motor:

- Rated Voltage: 12Vdc.
- No Load Speed: 50 r/min.
- No Load Current: 140mA.
- Load Torque Current (With Load): 800mA.
- Load Torque Speed (With Load): 36 r/min.
- Torque: 6.3 Kgf.cm (0.62 N.m).
- Output Power: 2.3 W.
- Stall Current: 3000mA.
- Motor Weight: 350~400 gm.

In figure 3.5

Figure 3.5: **DC Motor 12vdc**

- **Kit L298 Board**

The L298 Stepper Controller makes it easy to drive either two DC motors or another output. This is a very high-quality board and is very compact for designs where space really matters.

Features:

- Double H bridge drive
- Chip L298N (ST NEW)
- Logical voltage 5V
- Drive voltage 5V-35V
- Logic current 0mA-36mA
- Drive current 2A per channel (MAX single bridge)
- Max power 25W
- Weight 30g
- Small size 43*43*27mm (approx. 1.75" x 1.75" x 1")

In figure 3.6

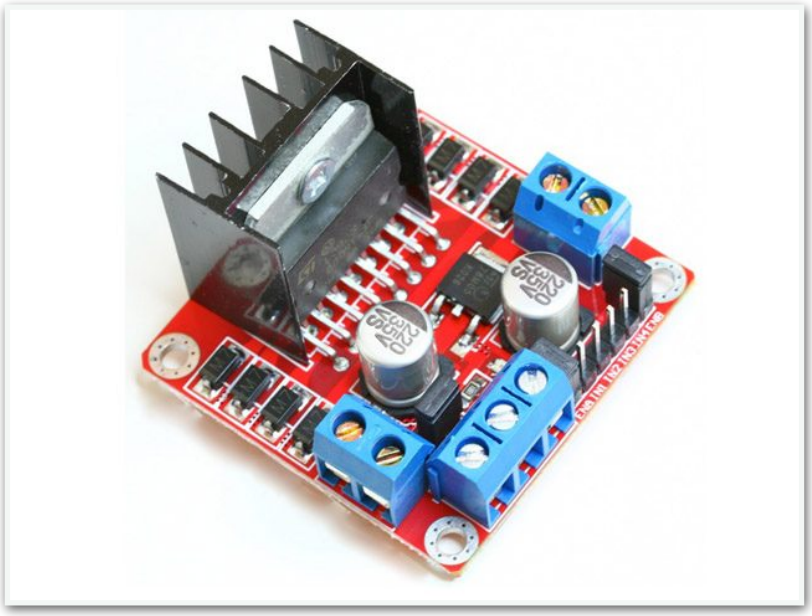


Figure 3.6: Kit L298 Board

- **LCD module:**

Features of 16×2 LCD module.

- Operating Voltage is 4.7V to 5.3V.
- Current consumption is 1mA without backlight.
- Alphanumeric LCD display module, meaning can display alphabets and numbers.
- Consists of two rows and each row can print 16 characters.
- Each character is built by a 5×8-pixel box.
- Can work on both 8-bit and 4-bit mode.
- It can also display any custom generated characters.
- Available in Green and Blue Backlight. ***In figure 3.7***

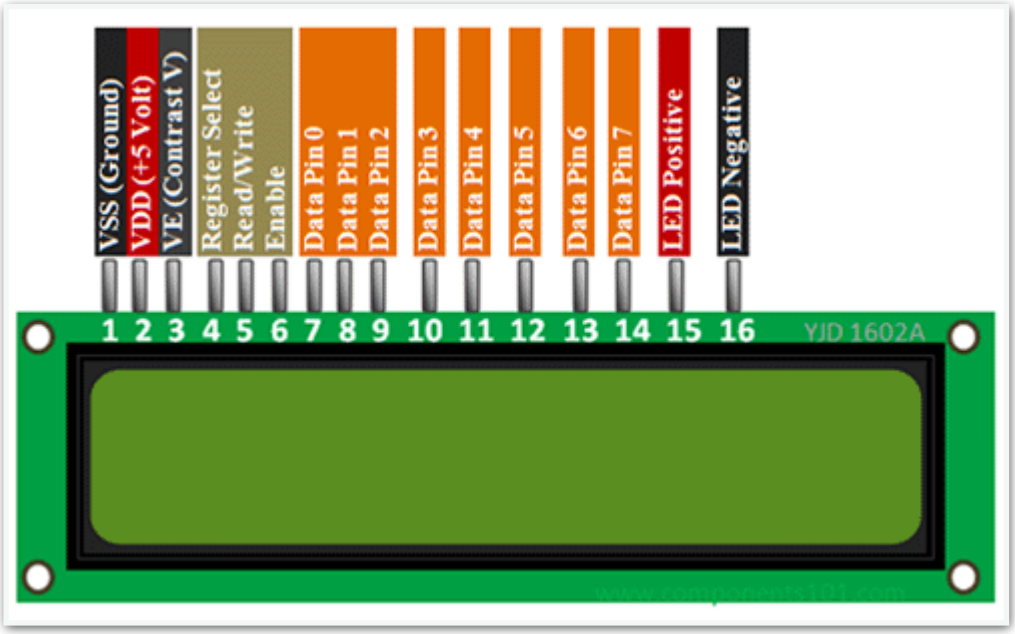


Figure 3.7: **LCD 2*16**

3.2 Project Schema

The control system of the smart baby cradle is equipped with

- power source 5 Vdc for the raspberry pi and then all components connected to it such as sound sensor module, temperature and humidity sensor, camera module, speaker.
- power source 12vdc that supplies to the DC motor and fan through Kit L298 Board

1) Raspberry pi with DC Motor:

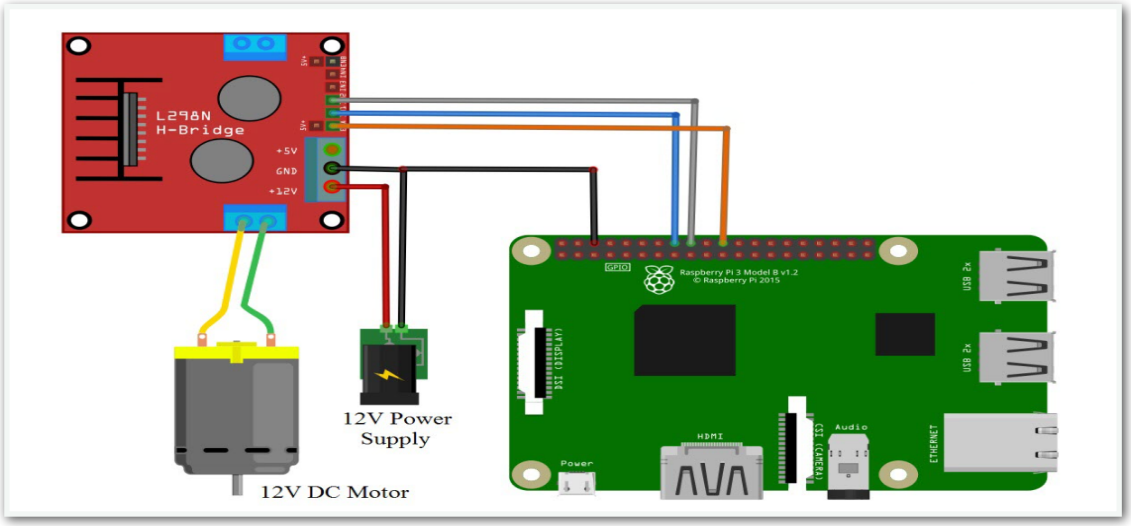


Figure 3.8: Raspberry Pi with DC Motor

first power source 12vdc connected to the kit that have two outputs.one of them connected to dc motor to control the speed of it and another output connected to the fan and another schema when we added fan to the motor with the power supply 12 vdc and drive to control of them through Kit L298 Board that have two output connected.

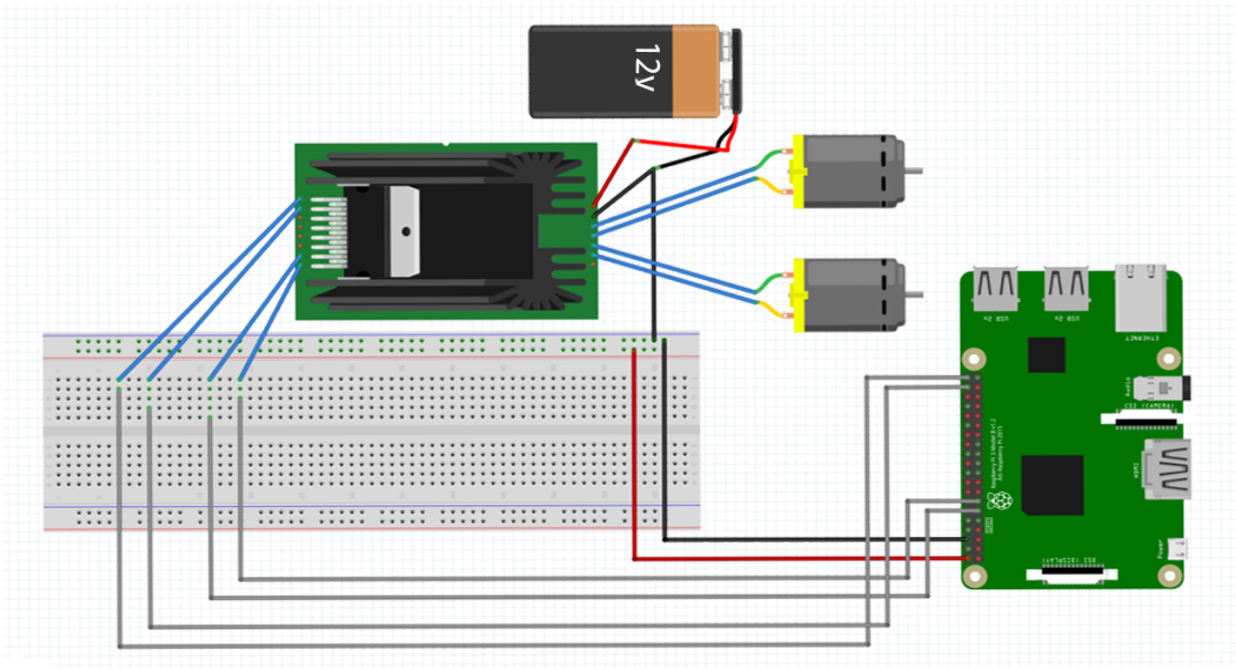


Figure 3.9: Motor & Fan with Power Supply

2) Live Audio and Video Streaming:

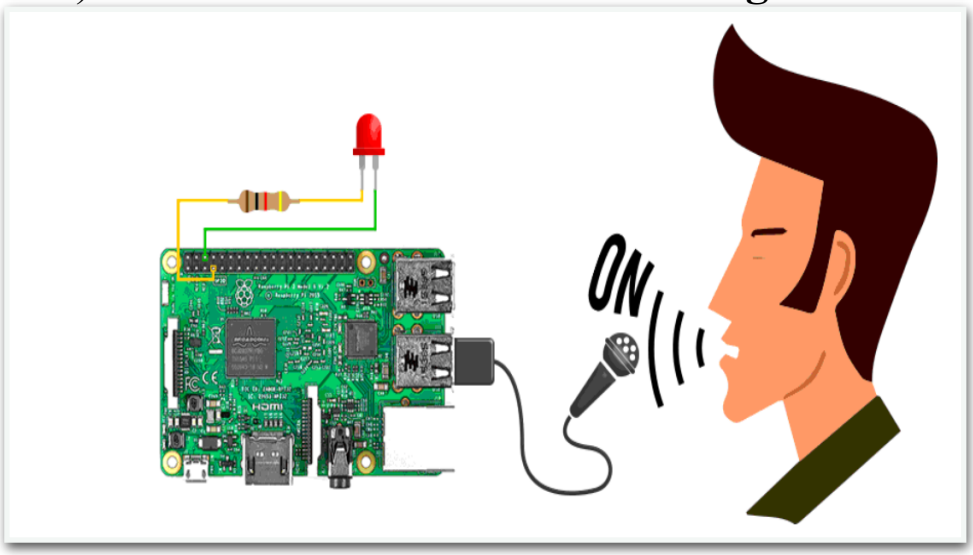


Figure 3.10: Audio & Video Stream

Since the Raspberry Pi device does not support a built-in mic or camera, we will use a Mini USB Microphone and a camera to plug them with the Raspberry Pi. We use a camera module that does not employ an infrared filter so it gives the ability to take

pictures in the darkness using infrared lighting in addition to the daylight pictures.

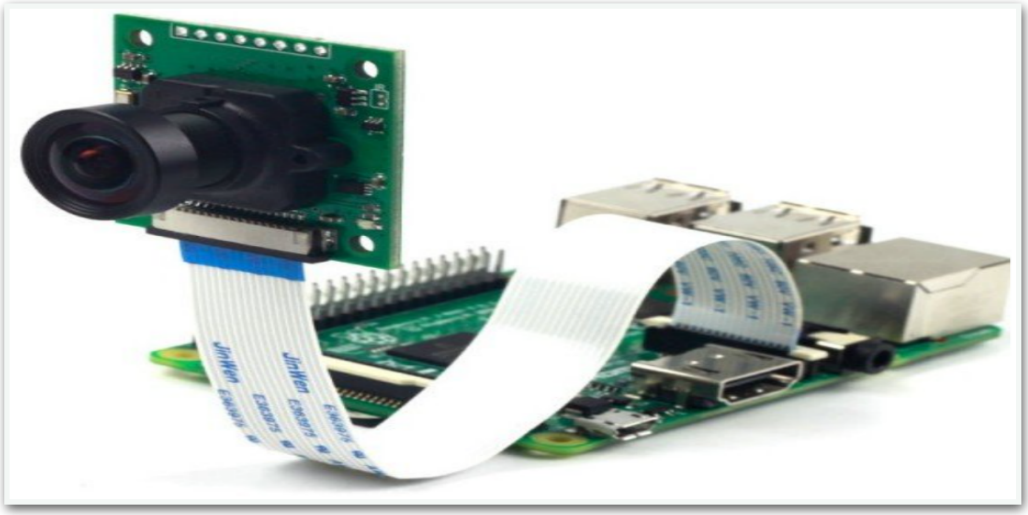


Figure 3.11: Camera Module

3) Temperature and Humidity Features

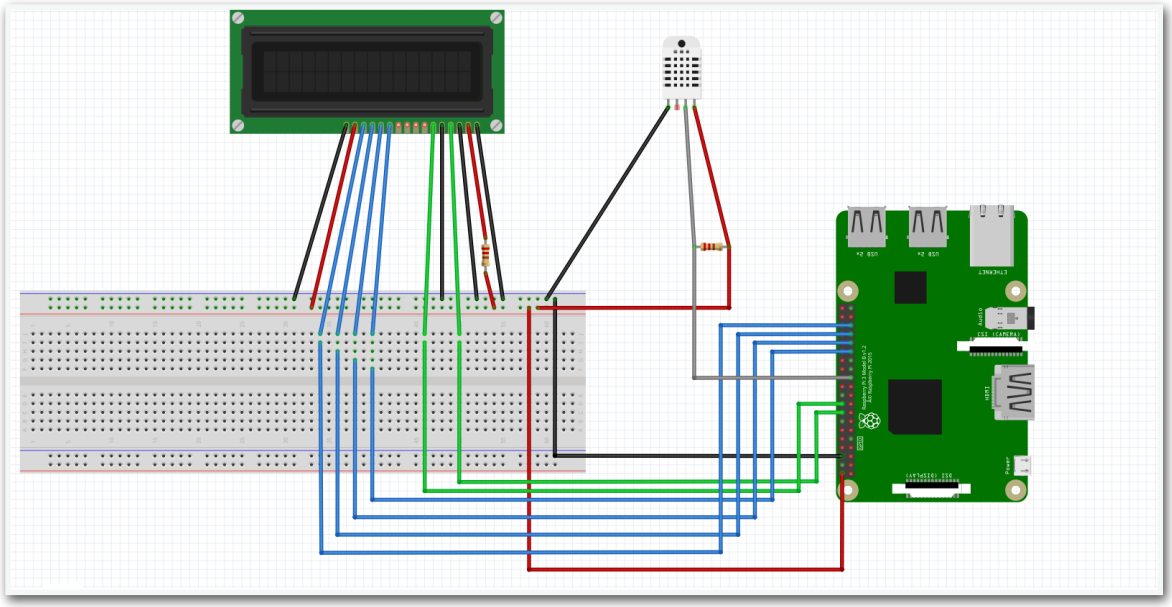


Figure 3.12: Temperature & Humidity with Raspberry Pi

Ultra-low-cost sensors for humidity and temperature are attached with the Raspberry Pi and LCD module for measuring the

surrounding temperature value and humidity change and display on the smart cradle. Firebase Real-time Database stores and synchronizes the measured data with NoSQL cloud database, as shown in. The stored data format is based on JavaScript Object Notation (JSON) and can be sent to the client within Real-time.The components are installed on the breadboard to ensure that the system works successfully before transferring the components onto the solder circuit panel.

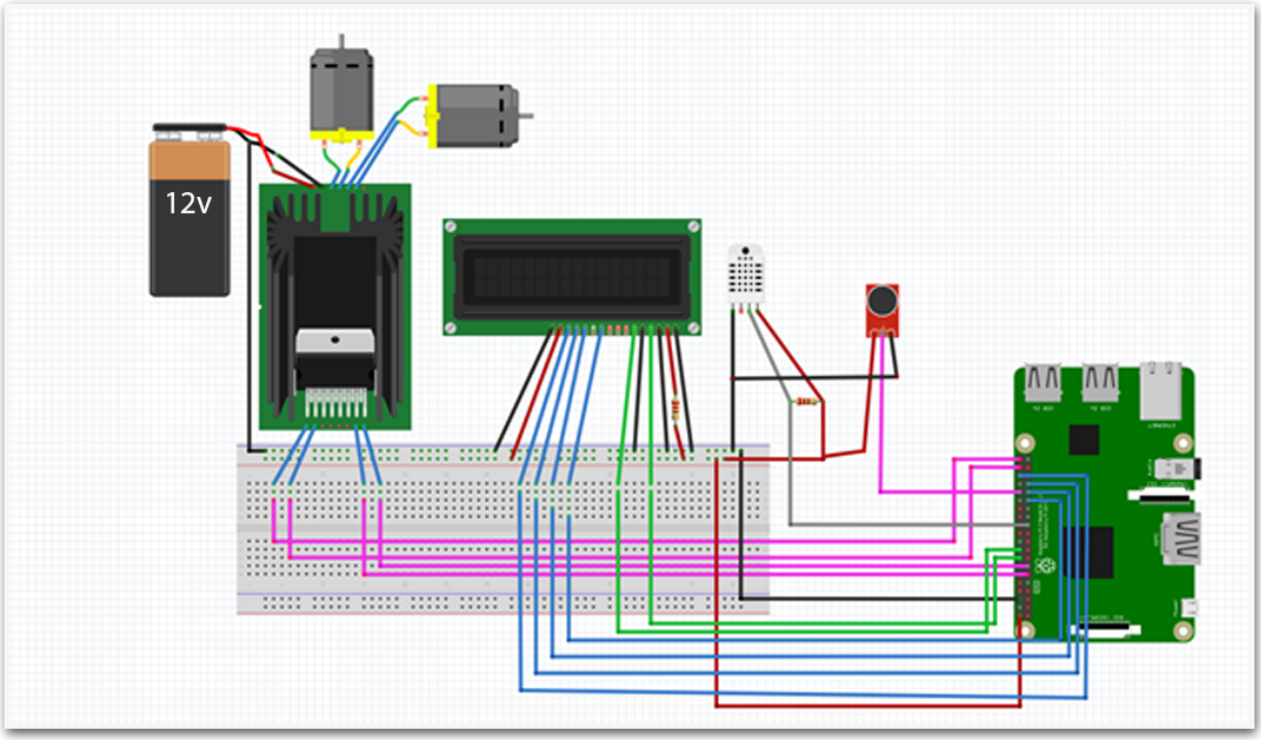


Figure 3.13: All Components with Raspberry pi

3.3 Baby Cradle Design

Every product needs a complete design with exact measurement before it can be fabricated. The design plays a fundamental role in this project. Therefore, prior to the design process, a few criteria were considered. We visualized our entire sketch by using a more practical picture. For this baby cradle, we designed the cradle dimen-sions and the main components, which allowed swinging and attachment of the developed monitoring system.

The baby cradle is rectangular in shape and has fences that keep the baby from falling off the front and side. The baby can still look outside from their cradle. The back side is similar to a door, but it does not operate sideways and it moves to 90 degrees from the initial state.This design is also suitable for babies 1-2-year-old and younger.Value-added processes, including cutting, paneling, drilling, sawing, forming and machining, were performed in the workshop.

The underlying principle involved cleaning and flattening the wood surface of the cutter mark to obtain a flat surface and **90°** of the four sides of the wood. Material selection is a core step in the process of designing any physical object. The systematic selection of the best material for the given application begins with the properties and costs of candidate materials. Selecting the appropriate material is one of the keys that lead to success.

Figure 3.14: Cradle from the Above

Figure 3.15: Cradle from the Front

This baby cradle has a rectangular shape, and the fences consist of five small bars that will prevent falls, but still allow the baby to look out from the cradle.

Figure 3.16: Cradle's Holder

The cylinder on top at the holder of the cradle is the baby's camera that is placed on top of the baby to monitor the baby's condition.

Figure 3.17: Painted Cradle

It has a curved shape, indicating that the base is smaller size than the upper part. Furthermore, the baby cradle has a classic design, when compared with some previous designs.

Figure 3.18: Motor Place

The cradle is designed to connect to the shaft of the geared motor. It swings when the switch is turned on by the user or when the sound sensor detected the baby's cry.

Figure 3.19: DC Motor

The rotating mechanism for swinging the cradle is displayed *in figure (3.20, 3.21).* A piece of aluminum sheet was cut into round shape.

Figure 3.20: Aluminum Sheet Before Cut

Figure 3.21: Aluminum Sheet After Cut

The DC motor was then locked with a motor coupler attached to the piece of the round aluminum sheet. The edge of the aluminum sheet was punched into 3 holes using a drill machine to allows us to change the cradle movement distance and the speed of the motor.

Then a small part of the aluminum sheet was cut from aluminum sheet and used to reduce the friction and noise resulting from the movement of the aluminum skewer with the movement of the motor shaft and the feeling of calm during the movement of the bed without any sound only the sound of the baby to facilitate the process of knowing it.

Figure 3.22: Aluminum Sheet with Motor Shaft

An Aluminum skewer was cut into 30 cm, was then screwed onto the edge of the aluminum sheet and with the cradle.

The motor is well installed on a piece of wood and below it is a piece of plastic to reduce the sound of its rotation and at a suitable height from the bottom, about 9 cents, to ensure the perfect rotation of the shaft 360 degrees without any obstacles.

Figure 3.23: Mechanism View 1

Figure 3.24: Mechanism View 2

This mechanism swings the baby cradle by raising the cradle to one side and to another side whenever the motor rotates. It causes
the cradle to swing by elevating around 20° on one side and dropping while the DC motor rotates to 360°.The shaft of the DC motor is connected to the cradle to allow swinging whenever the baby cry is detected or initiated by the user.

During the manufacturing process, the materials were constructed by combining the parts and components to obtain the finished product. The fabrication concept always involved the process of assembly.

The fabrication started with shop drawings, including precise measurements. Then, the fabrication stage was completed, and finally, the installation of the final project was done.

Chapter 4

4. Web Server And Firebase Connections

4.1 Sensors Configuration & Connections Between Raspberry Pi And Firebase

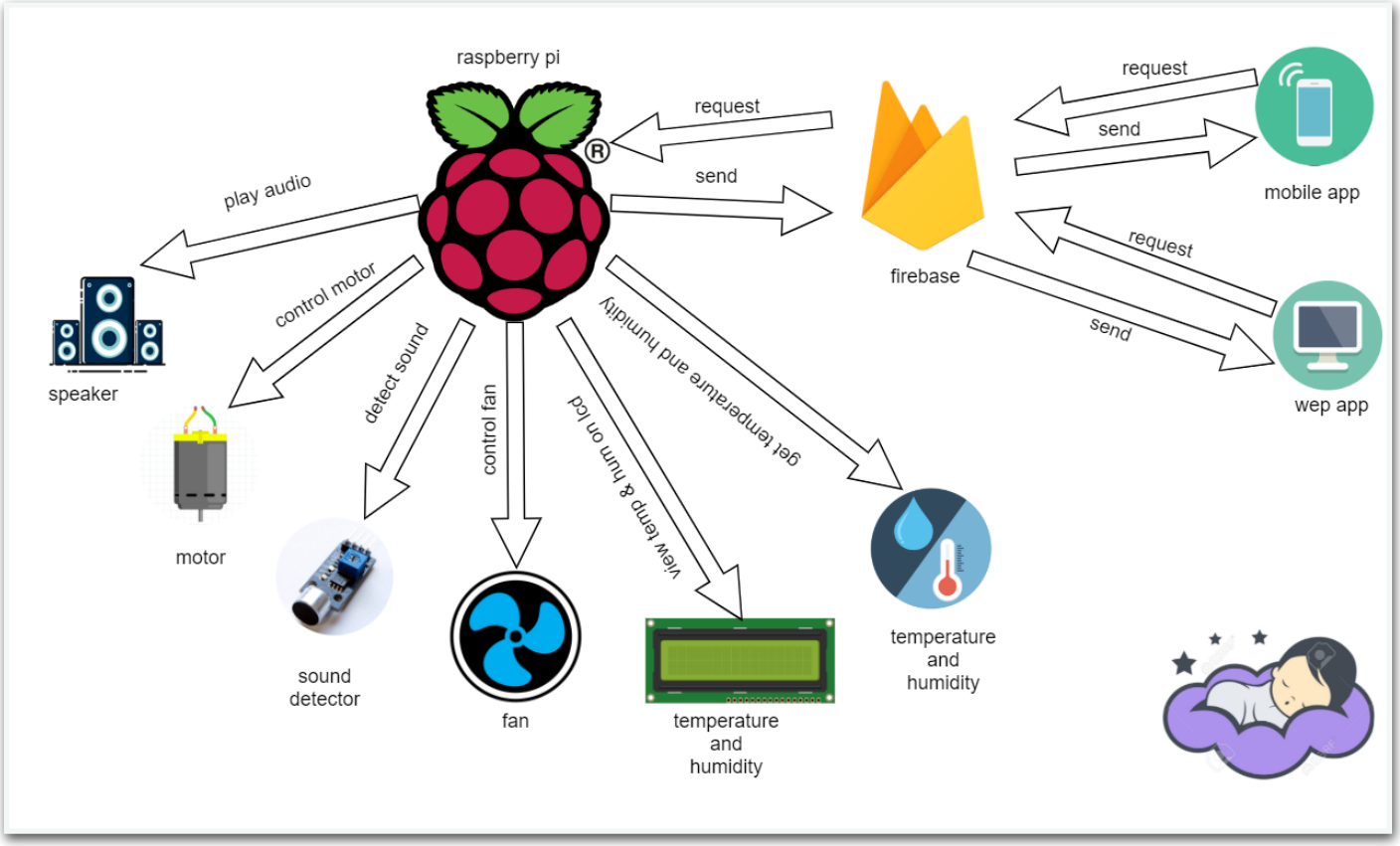


Figure 4.1: Firebase and Raspberry Pi Connection

4.1.1 Firebase Connection

Firebase is a mobile and web application development platform developed by Firebase, Inc. in 2011, then acquired by Google in 2014. As of October 2018, the Firebase platform has 18 products, which are used by 1.5 million apps.

Firebase is a **Backend-as-a-Service** (**BaaS**). It provides developers with a variety of tools and services that helps the developer to make web apps, mobile apps, or games regardless of server-side coding, API, or backend data storage. All the backed related stuff is managed by the Google cloud infrastructure. Firebase has some methods to connect with these services and you have to use them to make beautiful apps and websites.

In short, Firebase is a platform that allows you to build web and mobile applications without a server-side programming language.

Some firebase Basic Products and their Usage area:

- **Authentication** — user login and identity
- **Realtime Database** — real time, cloud hosted, NoSQL database
- **Cloud Messaging** — send messages and notifications to users
- **Cloud Firestore** — realtime, cloud hosted, NoSQL database
- **Cloud Storage** — massively scalable file storage
- **Cloud Functions** — "server-less", event driven backend
- **Firebase Hosting** — global web hosting
- **ML Kit** —SDK for common ML tasks
- **Analytics** — understand your users, and how they use your app
- **Predictions** — apply machine learning to analytics to predict user behavior
- **Remote Config** — customize your app without deploying a new version, monitor the changes

Installing pyrebase which is A simple python wrapper for the Firebase API. ***In Figure 4.2***

sudo apt-get install python3

sudo apt-get install python3-pip

sudo apt-get install python-dev

sudo pip install pyrebase

```
import pyrebase
config = {
  "apiKey": "AIzaSyDhinRkAu5k-3aL83EIe_thcTwhmu1fVvU",
  "authDomain": "baby-156b1.firebaseapp.com",
  "databaseURL": "https://baby-156b1.firebaseio.com",
  "storageBucket": "baby-156b1.appspot.com",
   "serviceAccount":  "firebase.json"
}
firebase = pyrebase.initialize_app(config)
db = firebase.database()
```

Figure 4.2: Json file from firebase to make a connection between

1- Motor Configuration and Connection with Firebase:

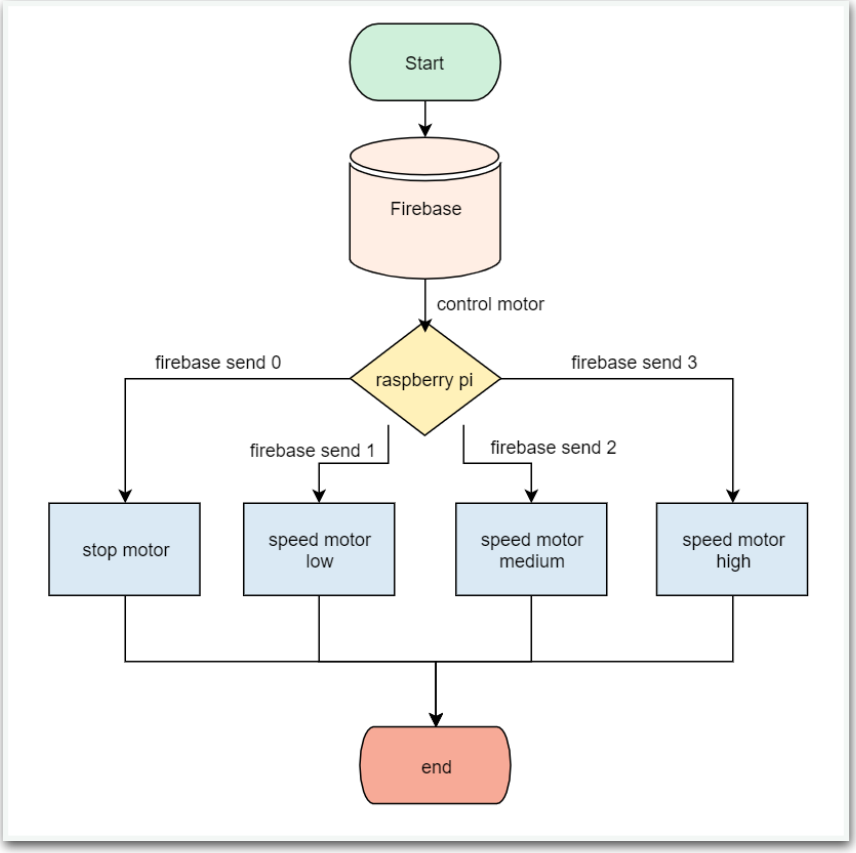


Figure 4.3: Motor Diagram operation

```
in1 = 17
en1 = 27
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(en1,GPIO.OUT)
GPIO.setup(in1,GPIO.OUT)
p1=GPIO.PWM(en1,1000)
p1.start(50)

while True:
    control1 = db.child("Motor/run").get()
    level1 = db.child("Motor/level").get()

    if (control1.val()==0):
        GPIO.output(in1, 0)
    elif (level1.val()==1):
        p1.ChangeDutyCycle(50)
        GPIO.output(in1, 1)
    elif (level1.val()==2):
        p1.ChangeDutyCycle(75)
        GPIO.output(in1, 1)
    elif (level1.val()==3):
        p1.ChangeDutyCycle(100)
        GPIO.output(in1, 1)
```

Figure 4.4: Motor Code

2- Fan Configuration and Connection with Firebase:

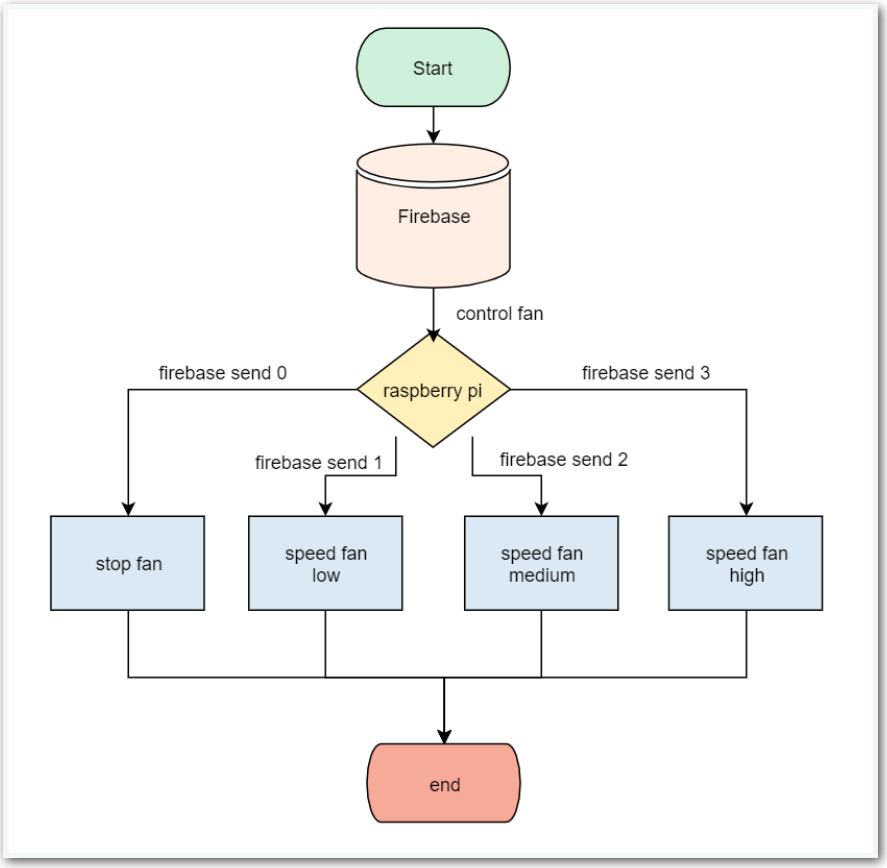


Figure 4.5: Fan Diagram operation

```
in2 = 20
en2 = 21
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(en,GPIO.OUT)
GPIO.setup(in1,GPIO.OUT)
p2=GPIO.PWM(en,1000)
p2.start(50)

while True:
    control2 = db.child("Fan/run").get()
    level2 = db.child("Fan/level").get()

    if (control2.val()==0):
        GPIO.output(in2, 0)
    elif (level2.val()==1):
        p2.ChangeDutyCycle(50)
        GPIO.output(in2, 1)
    elif (level2.val()==2):
        p2.ChangeDutyCycle(75)
        GPIO.output(in2, 1)
    elif (level2.val()==3):
        p2.ChangeDutyCycle(100)
        GPIO.output(in2, 1)
```

Figure 4.6: Fan Code

3- DHT 11 sensor and LCD:

Download Adafruit Python DHT Sensor Library to capture reading of DHT 11 sensor

git clone: https://github.com/adafruit/Adafruit_Python_DHT.git

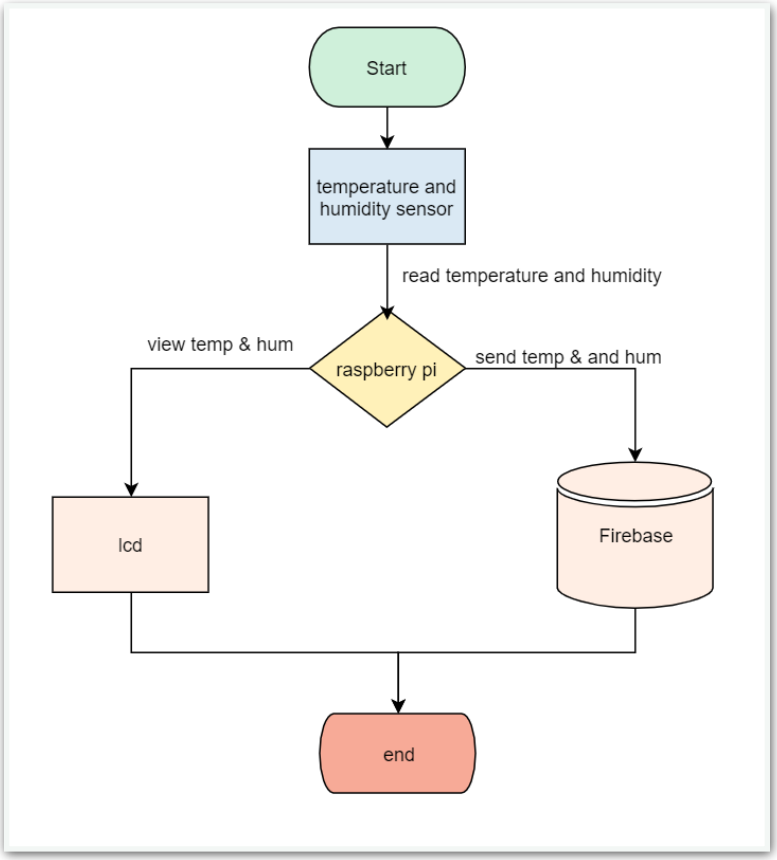


Figure 4.7: Lcd and DHT11 Diagram

```
while True:

    humidity, temperature = Adafruit_DHT.read_retry(11, 4)
    print 'Temp: {0:0.1f} C  Humidity: {1:0.1f} %'.format(temperature, humidity)

    # Send some test
    lcd_string("IOT Baby",LCD_LINE_1)
    lcd_string("Monitor",LCD_LINE_2)
    time.sleep(2)


    # Send some more text
    lcd_string(  'Temp= %.1f C' % temperature  ,LCD_LINE_1)
    lcd_string( 'Hum = %.1f %%' % humidity ,LCD_LINE_2)
    time.sleep(10)

    data = {"Humidity": ' %.1f ' % humidity,
            "Temperature": ' %.1f ' % temperature}
    db.child("status").set(data)
```

Figure 4.8: Lcd with DHT11 Code

4- Sound Detector:

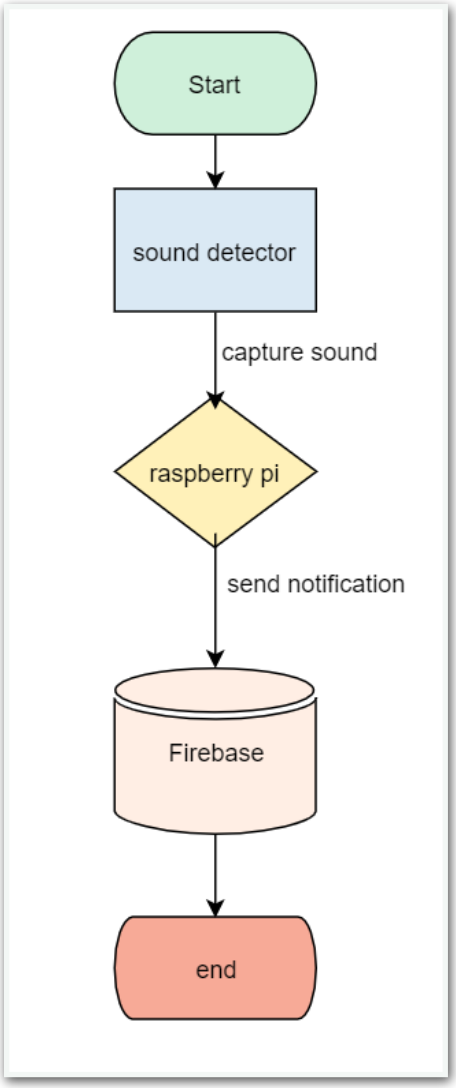


Figure 4.9: Sound Detection Operation

```
channel = 12
GPIO.setmode(GPIO.BCM)
GPIO.setup(channel, GPIO.IN)

def callback(channel):
    if GPIO.input(channel):
        print "Sound Detected!"
        db.child("Sound Detection/detected").set("yes")

    else:
        print "Sound Detected!"
        db.child("Sound Detection/detected").set("yes")

GPIO.add_event_detect(channel, GPIO.BOTH, bouncetime=300)
GPIO.add_event_callback(channel, callback)

while True:
    time.sleep(5)
    db.child("Sound Detection/detected").set("no")
```

Figure 4.10: Sound Detection Code

5- Play Songs:

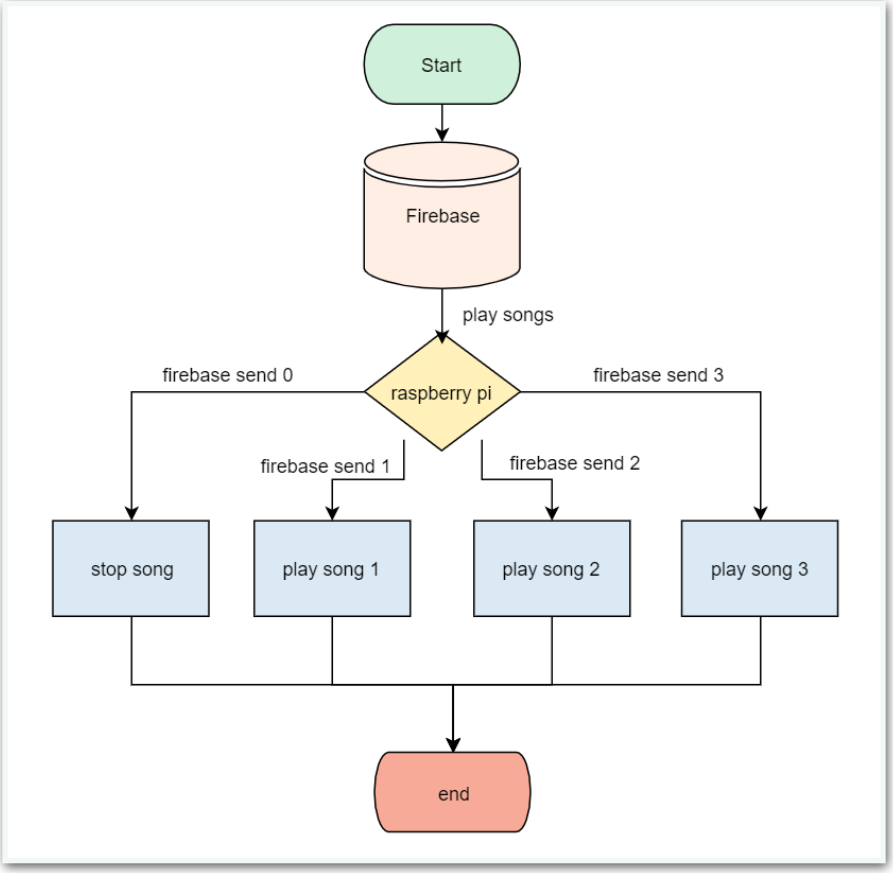


Figure 4.11: Playing Music Diagram

```
pygame.mixer.init()
s=pygame.mixer.music

while True:
    voice = db.child("Voices/voice").get()
    if (voice.val()==1):
        s.load("1.mp3")
        s.play()
        db.child("Voices/voice").set(5)
        if (voice.val()==5):
            pygame.mixer.music.get_busy()
    elif (voice.val()==2):
        s.load("2.mp3")
        s.play()
        db.child("Voices/voice").set(5)
        if (voice.val()==5):
            pygame.mixer.music.get_busy()
    elif (voice.val()==0):
        pygame.mixer.music.stop()
```

Figure 4.12: Playing Music Code

4.2 Live Streaming By UV4L Server

UV4L was originally conceived as a modular collection of ***Video4Linux2*-compliant**, cross-platform, **user space** drivers for real or virtual video input and output devices (with absolutely no external difference from kernel drivers). While still preserving the original intentions, UV4L has evolved over the years and now optionally includes a generic purpose *Streaming Server* plug-in, especially made for IoT devices, that can serve **custom** web applications that can make use of a number of **standard** and modern built-in services for *Real-Time Communications* such as encrypted, bidirectional data channels, audio and video streaming or conferencing over the web. ***In figure 4.13***

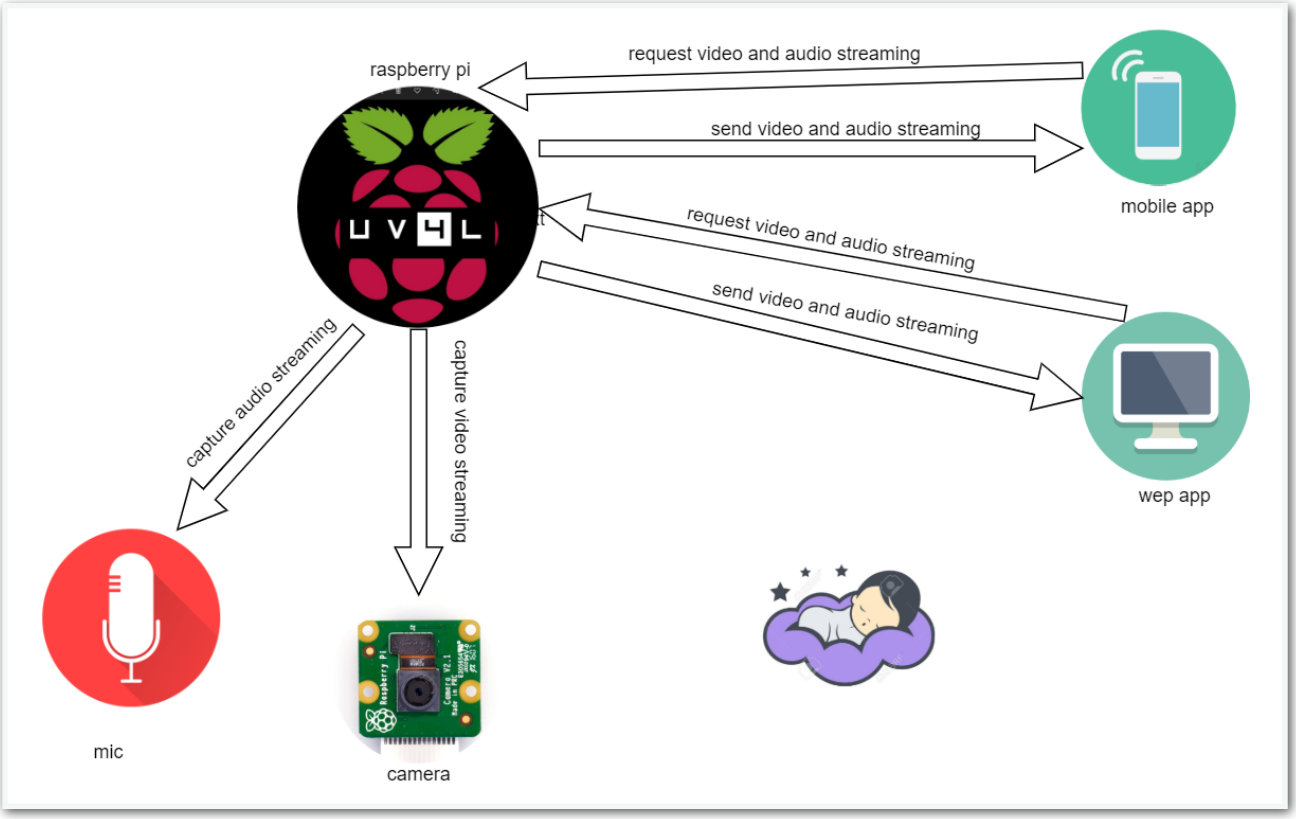


Figure 4.13: UV4L

4.3 Backend Server With Django Framework

4.3.1 Introduction To Django

Developing with web server (Django) A "web framework" offers a set of APIs for writing your own custom code in such a way that it can be called via the Web. Usually a framework will deal with common details such as HTTP header parsing, URL routing and so forth.
A web server is a piece of software which listens on a network port for incoming HTTP requests and responds to them.

Django makes it easier to build better Web apps more quickly and with less code.
Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Why Django?

With Django, you can take Web applications from concept to launch in a matter of hours. Django takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Ridiculously fast:
Django was designed to help developers take applications from concept to completion as quickly as possible.

Fully loaded:
Django includes dozens of extras you can use to handle common Web development tasks. Django takes care of user authentication, content administration, site maps, RSS feeds, and many more tasks — right out of the box.

Reassuringly secure:
Django takes security seriously and helps developers avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery and clickjacking. Its user authentication system provides a secure way to manage user accounts and passwords.

Exceedingly scalable:
Some of the busiest sites on the planet use Django's ability to quickly and flexibly scale to meet the heaviest traffic demands.

4.3.2 Install And Make Django Up And Work

○ Install Python

Being a Python Web framework, Django requires Python. See What Python version can I use with Django? for details. Python includes a lightweight database called SQLite so you won't need to set up a database just yet.

Get the latest version of Python at https://www.python.org/downloads/ or with your operating system's package manager.

You can verify that Python is installed by typing python from your shell; you should see something like. ***In figure 4.12***

```
Python 3.x.y
[GCC 4.x] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 4.12

○ Installing an official release with pip

This is the recommended way to install Django.

Install pip. The easiest is to use the standalone pip installer. If your distribution already has pip installed, you might need to update it if it's outdated. If it's outdated, you'll know because installation won't work.

```
$ python –m pip install Django
```

○ Verifying

To verify that Django can be seen by Python, type python from your shell. Then at the Python prompt, try to import Django:

```
>>> import django
>>> print(django.get_version())
3.0
```

```
$ python –m Django --version
```

If Django is installed, you should see the version of your installation. If it isn't, you'll get an error telling "No module named Django".

○ Create Project

If this is your first time using Django, you'll have to take care of some initial setup. Namely, you'll need to auto-generate some code that establishes a Django project – a collection of settings for an instance of Django, including database configuration, Django-specific options and application-specific settings.

4.3.3 Develop Django with our project

After installing Django, we will create a new project by using this command "django-admin startproject src"

"You can change "src" 'to any name you want'

This will create the following files.***In figure 4.13***

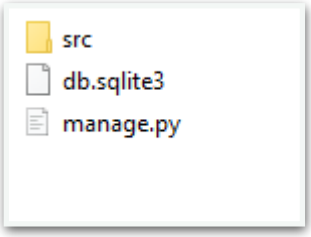
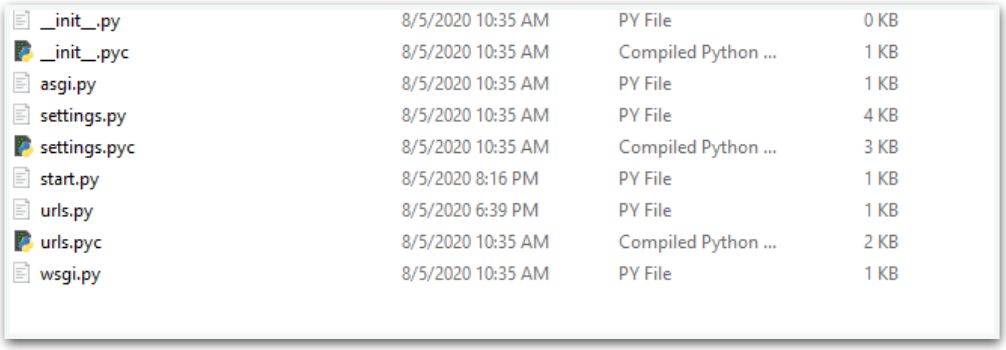


Figure 4.13

Second, run Django in the project folder:
by running this command "python manage.py run server".
This is the files of the server.***In figure 4.14***



__init__.py	8/5/2020 10:35 AM	PY File	0 KB
__init__.pyc	8/5/2020 10:35 AM	Compiled Python ...	1 KB
asgi.py	8/5/2020 10:35 AM	PY File	1 KB
settings.py	8/5/2020 10:35 AM	PY File	4 KB
settings.pyc	8/5/2020 10:35 AM	Compiled Python ...	3 KB
start.py	8/5/2020 8:16 PM	PY File	1 KB
urls.py	8/5/2020 6:39 PM	PY File	1 KB
urls.pyc	8/5/2020 10:35 AM	Compiled Python ...	2 KB
wsgi.py	8/5/2020 10:35 AM	PY File	1 KB

Figure 4.14

We will start to add our project files to the server's files by creating and editing the following files.***In figure 4.15***

1.a) create a new " start.py "file

```
# Threading Imports
import threading
from time import sleep
import time
from src.sensors.statusLcd import lcd
from src.sensors.sound import sound
from src.sensors.singlethread import singlethread

# Creating the single thread
singlethread = threading.Thread(target=singlethread, daemon=True)
singlethread.start()

# Creating the status thread
lcdStatusThread = threading.Thread(target=lcd, daemon=True)
lcdStatusThread.start()
#
soundThread = threading.Thread(target=sound, daemon=True)
soundThread.start()
```

Figure 4.15

1.b) create a new “ singlethread.py”file. *In figure 4.16*

```python
from sensors.motor import motor
from sensors.fan import fan
from sensors.play import music
def singlethread():
    # While server is running
    while(True):
        motor()
        fan()
        music()
    # Once server is closed
    # GPIO.cleanup()
```

Figure 4.16

2.Add the files of project into the server’s subfolder “sensors”

- **motor.py**
- **fan.py**
- **status lcd.py**
- **play.py**
- **sound.py**

After adding these files

We edit the “ursl.py” file like this. ***In figure 4.17***

```python
from django.contrib import admin
from django.urls import path
from . import start

urlpatterns = [
    path('admin/', admin.site.urls),
]
```

Figure 4.17

3.Now, we only need to run the server by typing this command in the path of our server's folder,

"python manage.py run server"

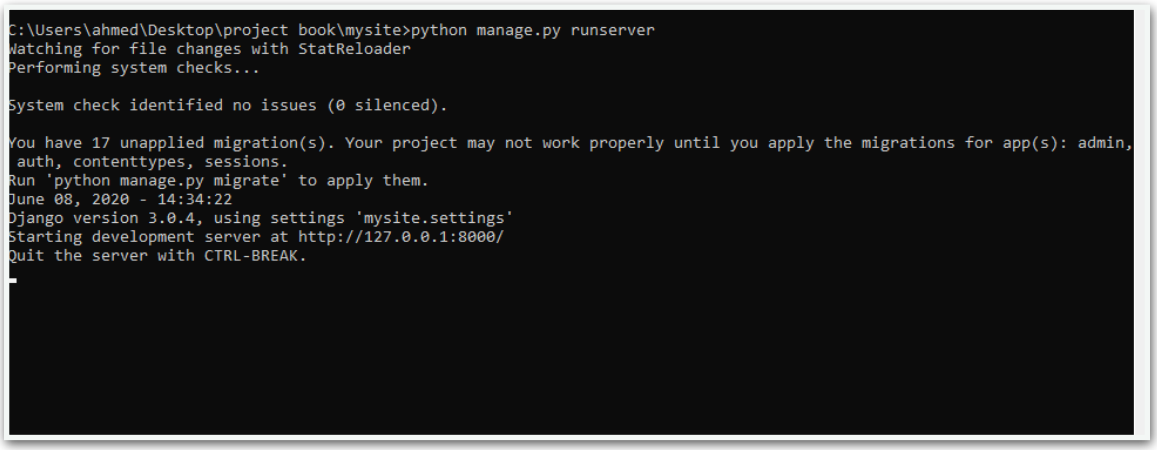


Figure 4.18

That means that the server and our threads is up and running with our project files. ***In figure 4.18***

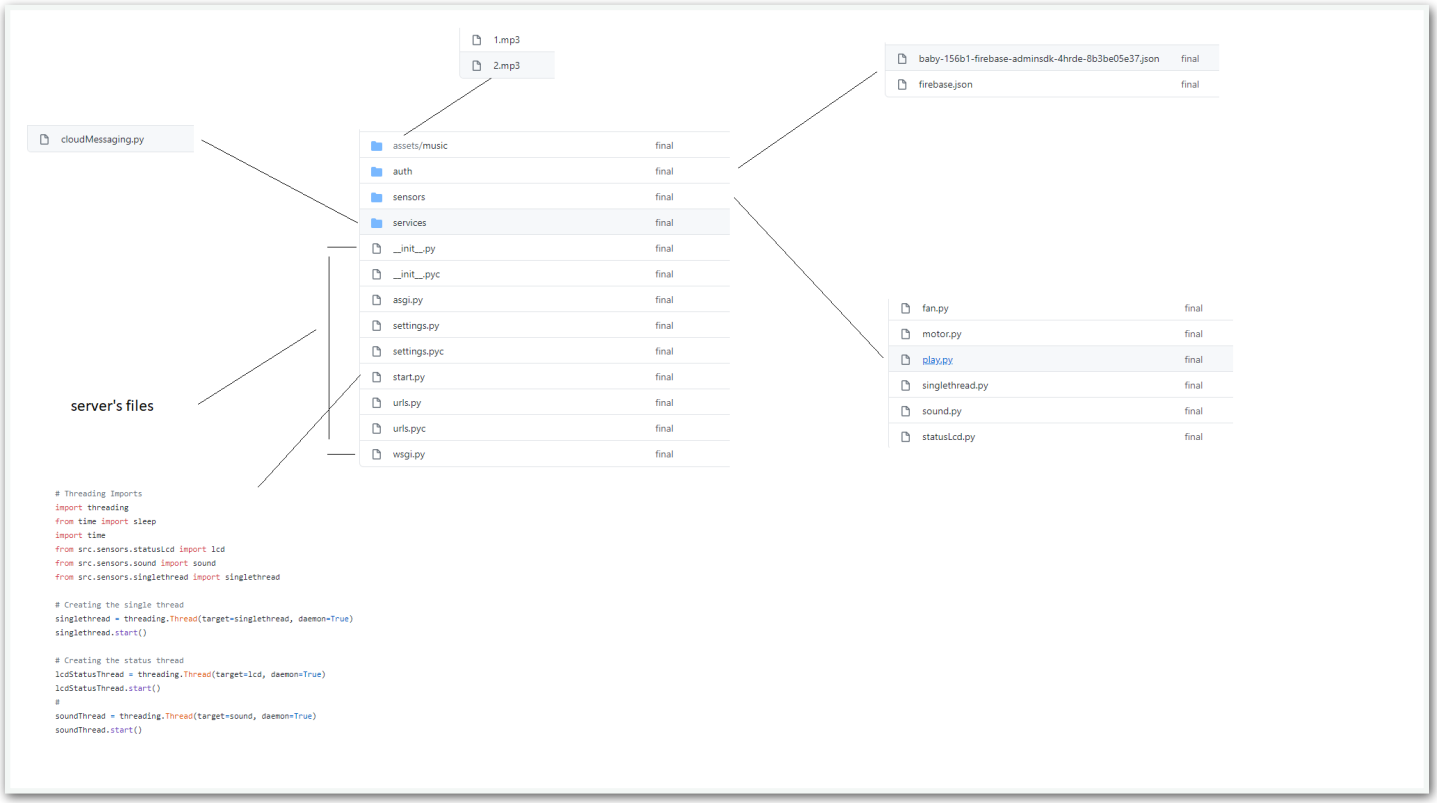


Figure 4.19: Server's Files Scheme

Chapter 5

5. Platform Applications

5.1 Sequence of Operation

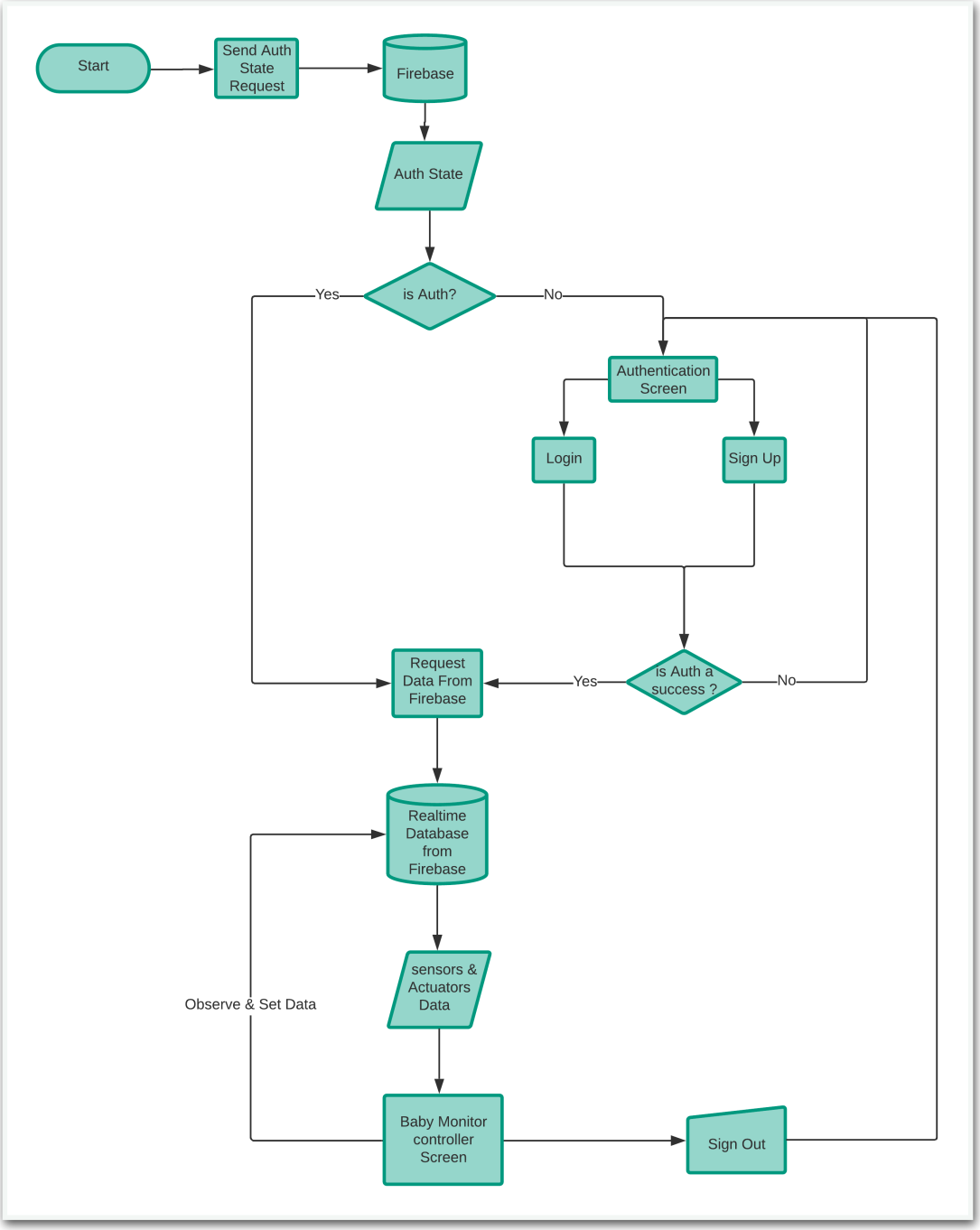


Figure 5.1: End To End Operation Diagram

End to End Operation of Application will be like this in that Diagram below. In figure 5.1

When the parent opens the application the authentication request will be sent to firebase authentication to check if that user is connected there, so if the user is already registered , it sends http request to firebase to fetch data in the Realtime database and it will shows the baby monitoring controller screen.

If the user doesn’t sign in yet it shows the Login Screen, the user has to enter his e-mail and password and it will send http request to check if the user is already registered in firebase if not will the user may be the new and he has to sign up , he will press sign up button to transfer to Sign Up Screen, he will enter new e-mail and new password and must enter Raspberry Pi ID device of Baby Cradle that the parents have bought to check it will be the same to not let anyone to make the account and control the cradle by the application . Then when the parent make a new account he will use the application, see the what's going on about baby and see him, listen to him , he can shake the cradle and play music for him, etc.

The data that set or fetch to/from firebase occurs immediately and listen for every registered, so the parents see the modified changes happens in the application.This the advantage of real-time database. In figure 5.2 , In figure 5.3

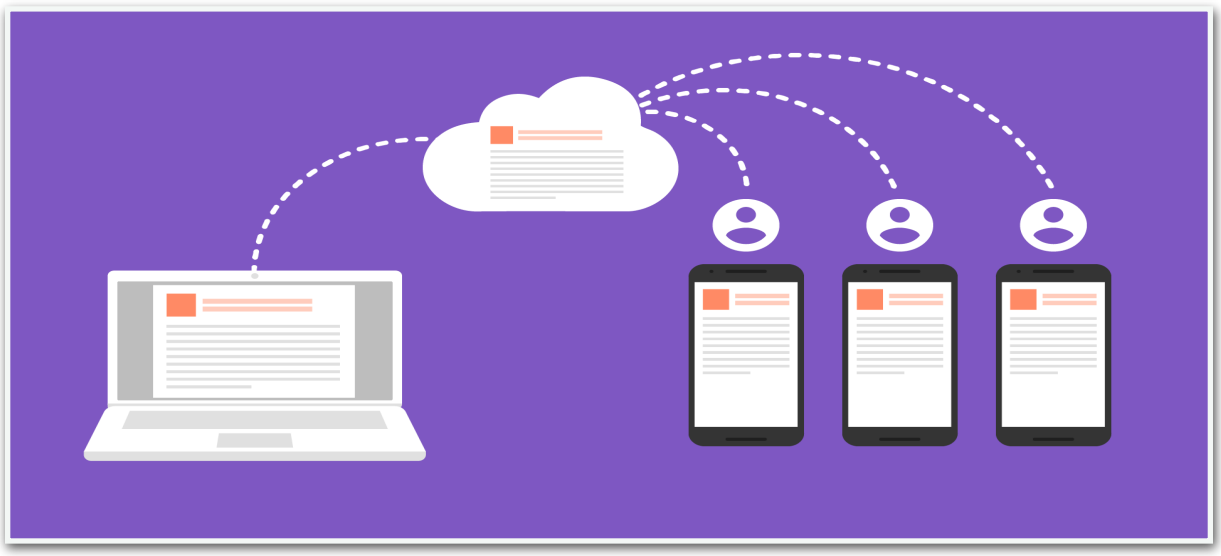


Figure 5.2: Realtime Database

Figure 5.3: Realtime Database in Firebase

In **Motor** Operation with firebase when the user opens the application it will send request data from firebase and listen for any changes that happens the firebase send him the data that stored in it , if it already runs or not by values (1/0) and degree level that set (Low-Medium-High).***in figure 5.4***

In **Fan** operation is the same of Motor. ***In figure 5.5***

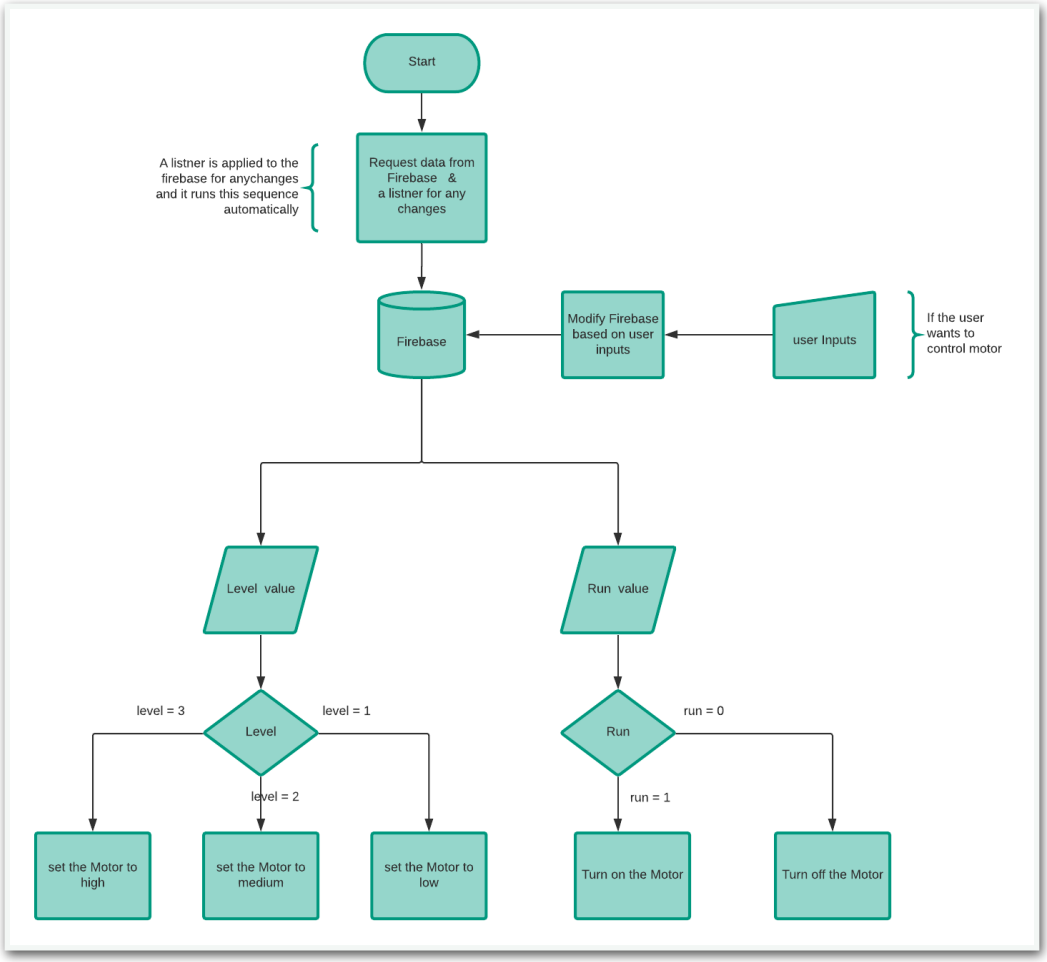


Figure 5.4: Motor Diagram

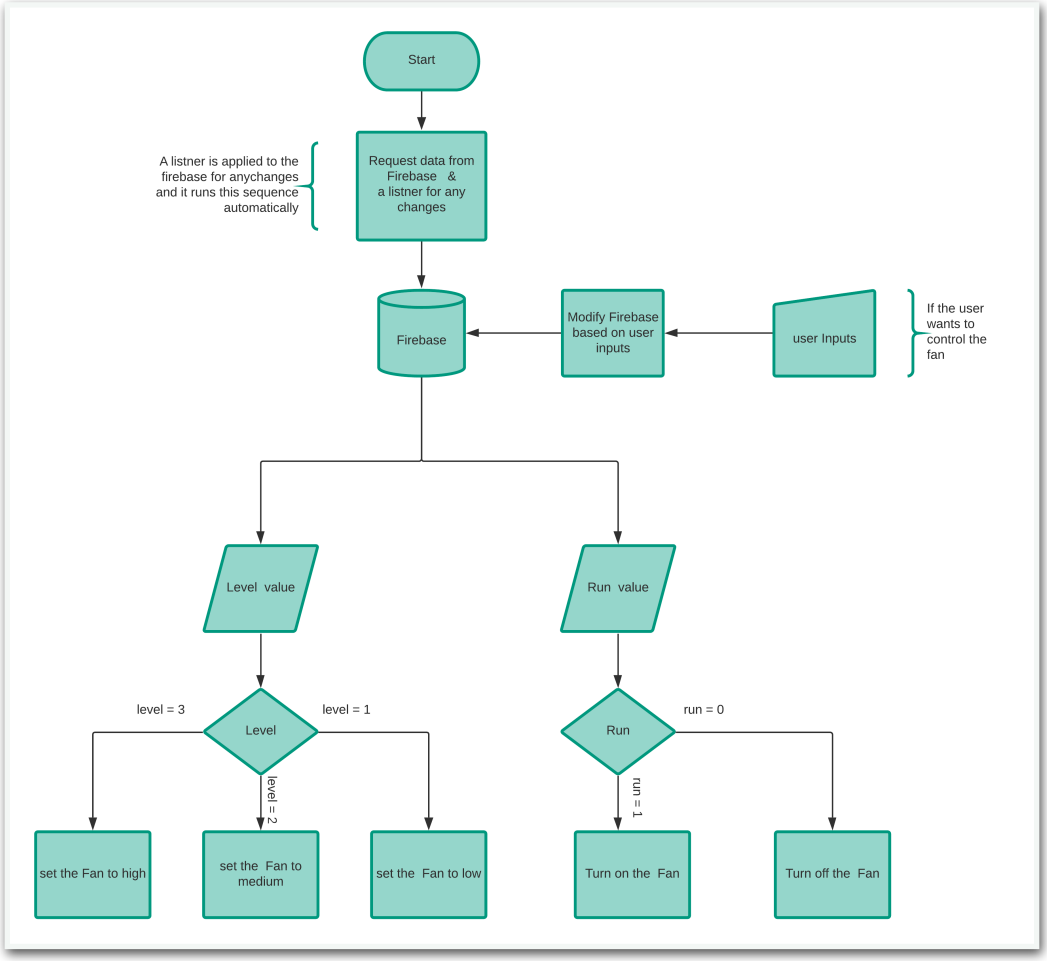


Figure 5.5: Fan Diagram

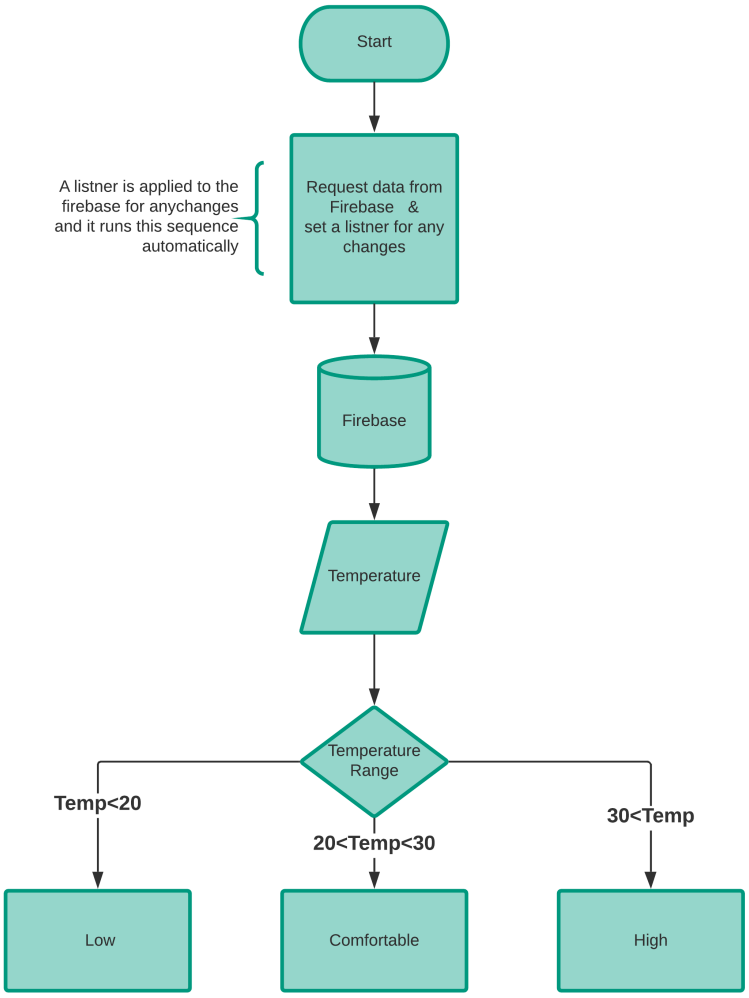


Figure 5.6: Temperature Diagram

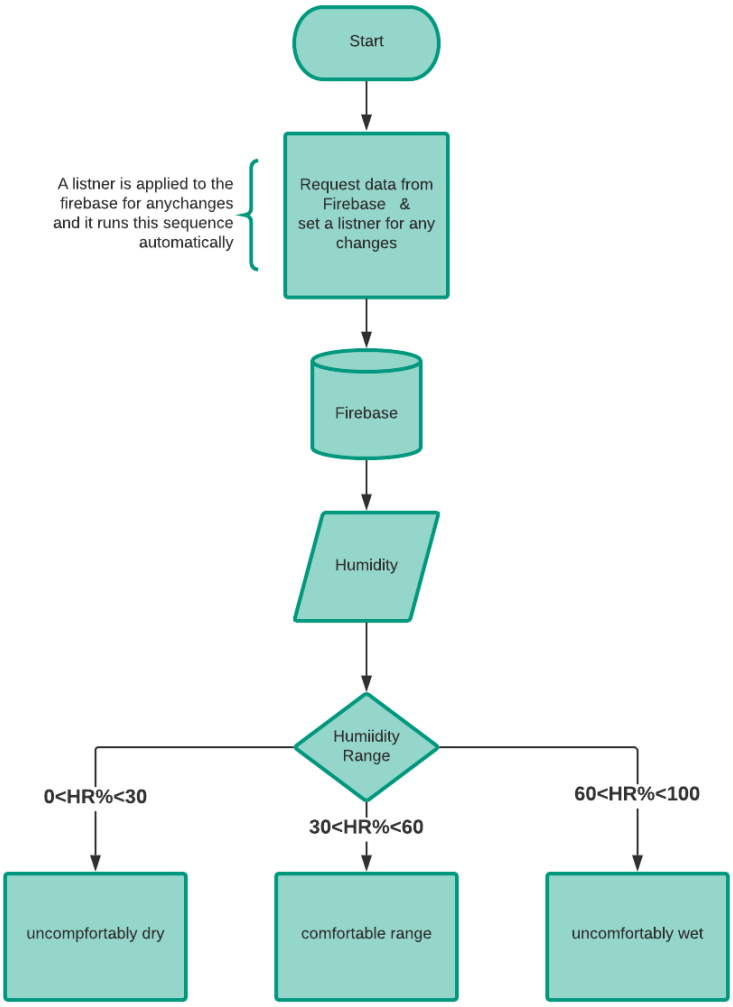


Figure 5.7: Humidity Diagram

In the Temperature and Humidity, In (figure 5.6, figure 5.7) Operation just fetches the data from firebase and are listening for any change that happens, then each of them shows the result of value and renders the color reflection depends on the value confined in the range.

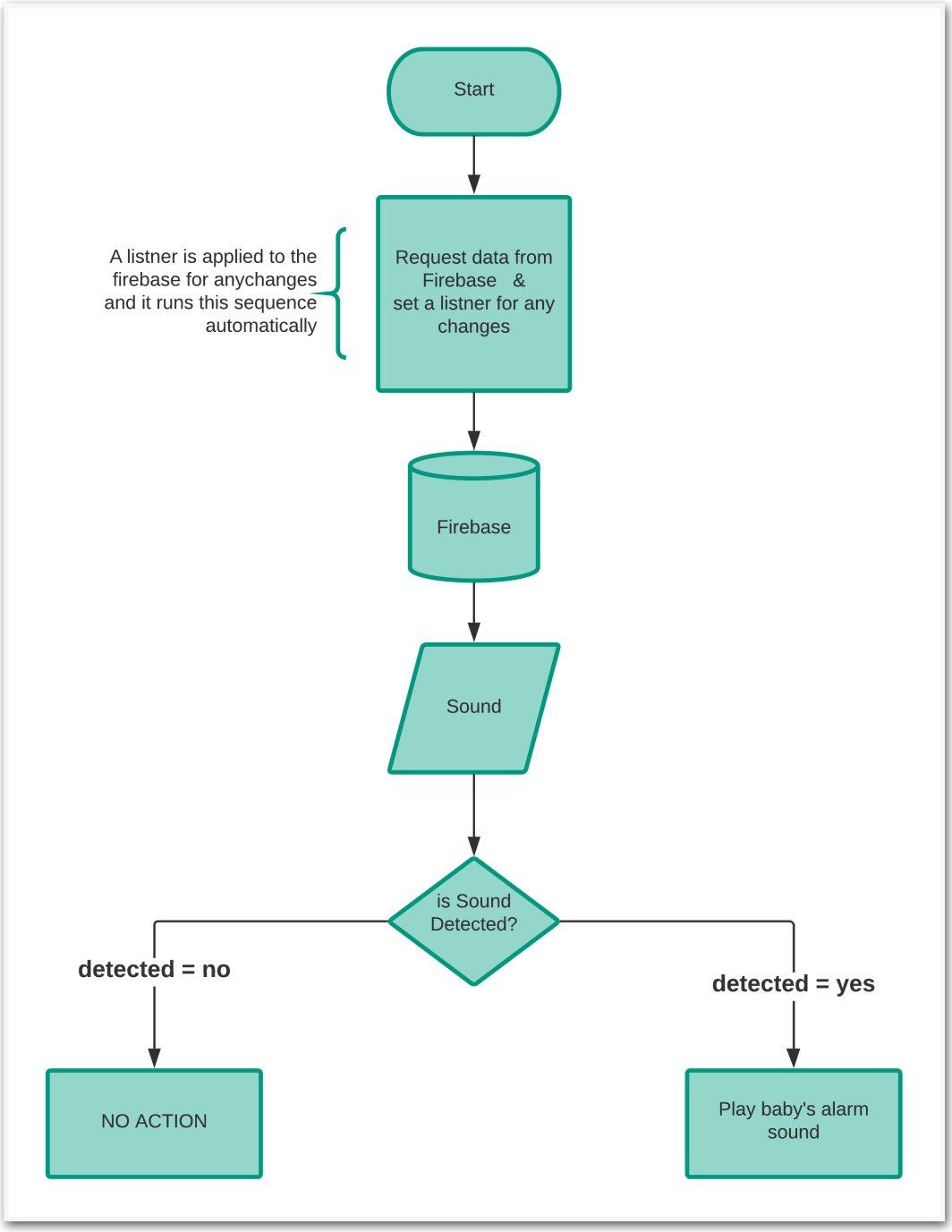


Figure 5.8: Sound Detection Diagram

In **Sound Detection** Operation,In figure 5.8, listens any change high frequency specifically from sound detection sensor if the baby cries the sensor will detect it, and send data to the firebase and its value will be “yes” as detected.

When sensor detected Baby's Crying sound, it will send http request to firebase and depends on this data it will run motor on automatically if it weren't enable, then the cradle will shake a period of time till baby will calm down. ***In figure 5.9***

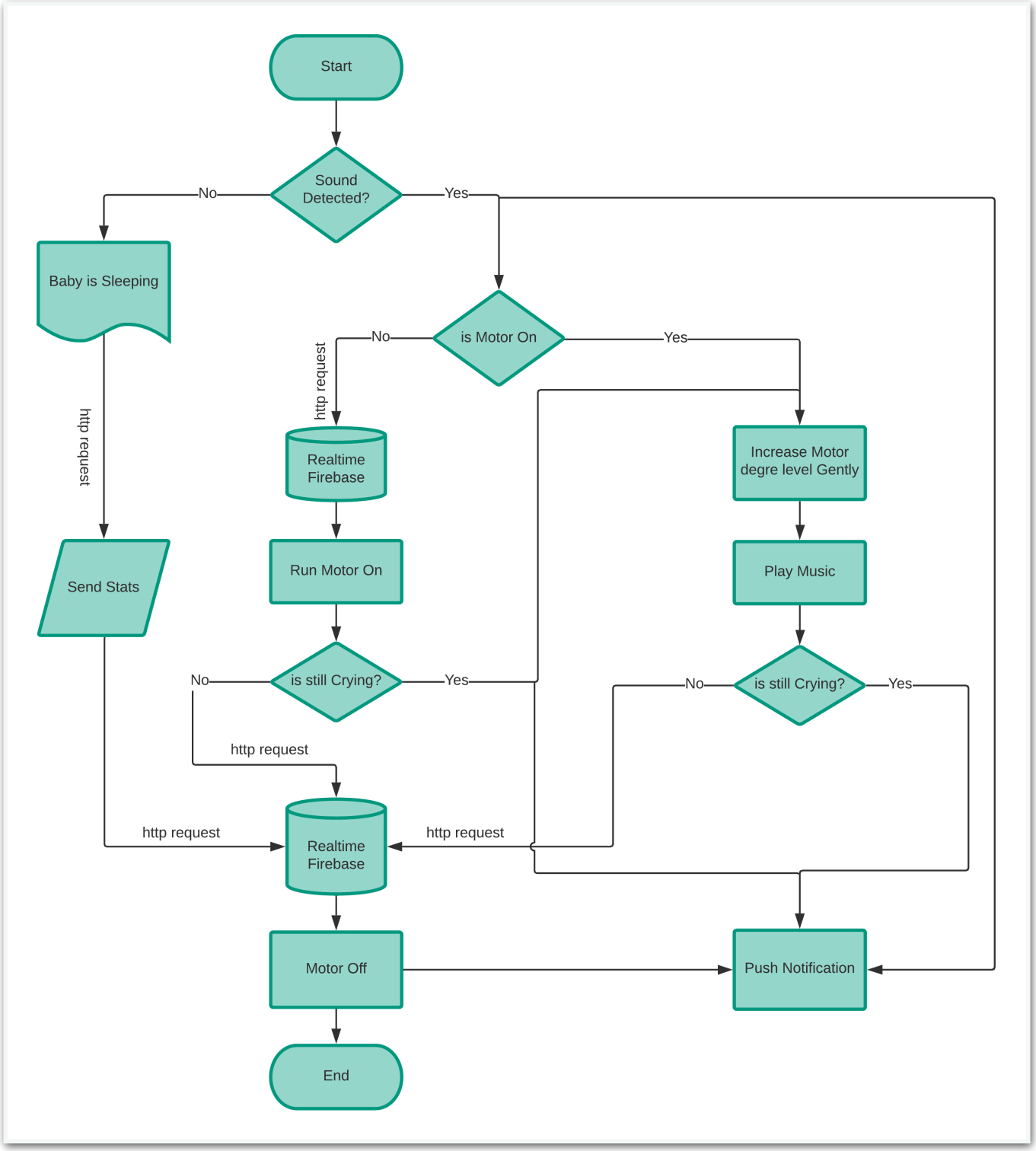


Figure 5.9: Auto Shaking Diagram

When temperatures degree is greater than 30˚ C , it will send http request to firebase to notify it that the temperature is high, so it will turn Fan on automatically , and if the temperature is still the same degree above 30˚ C, it will increase the levels of fan gradually till the baby feels comfortable temperature. ***In figure 5.10***

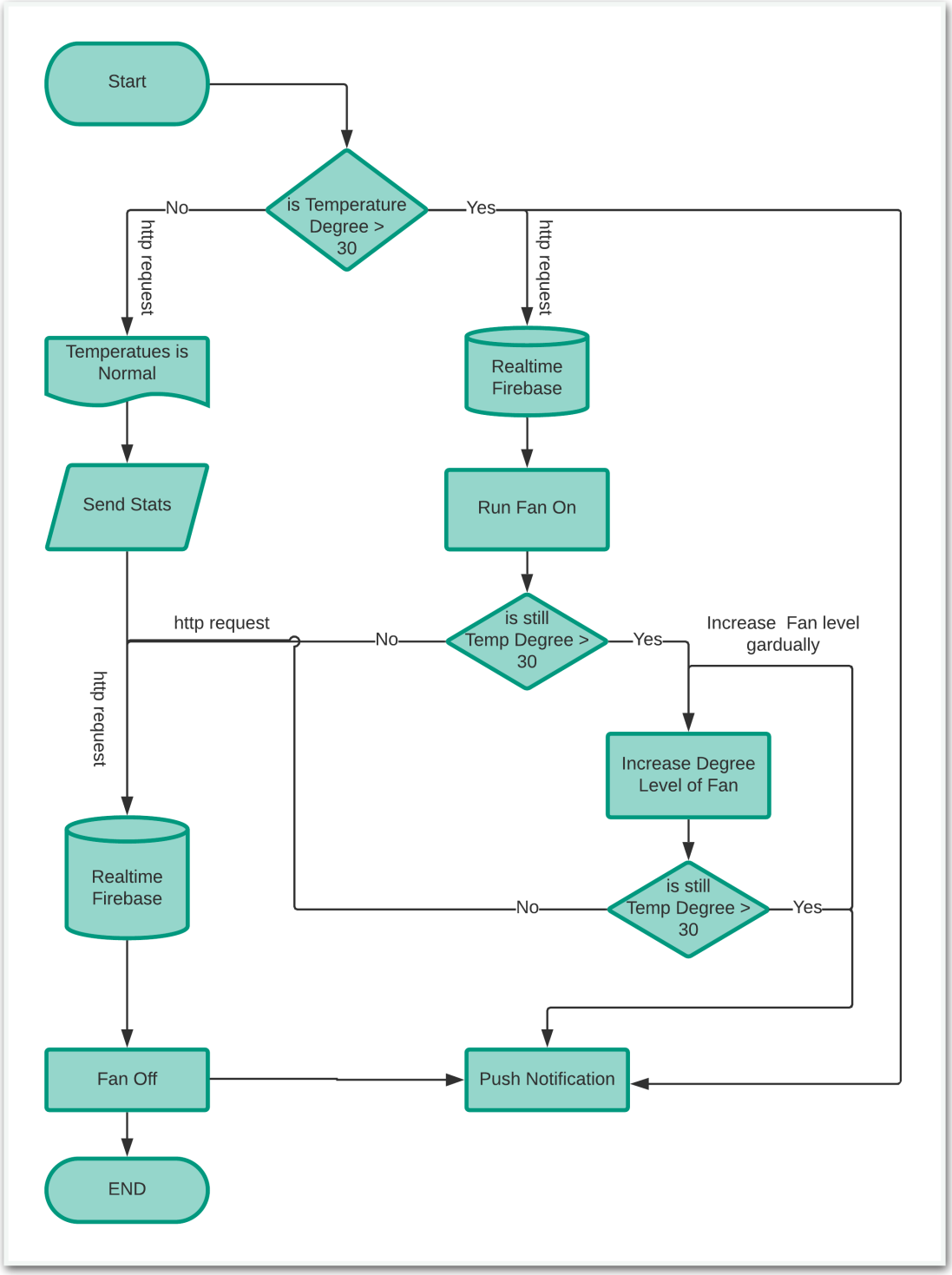
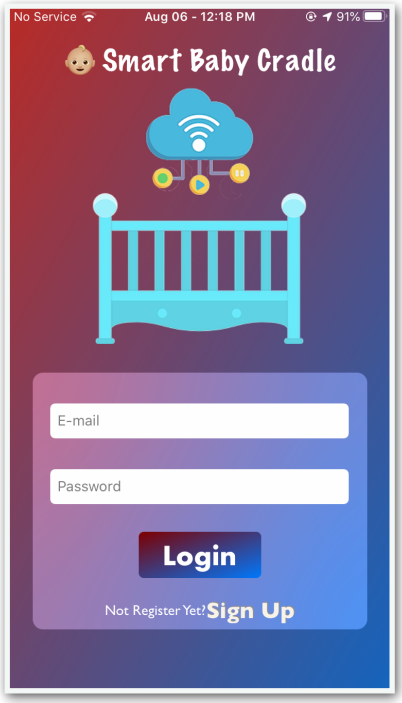


Figure 5.10: Auto Fan Diagram

5.2 Software Development

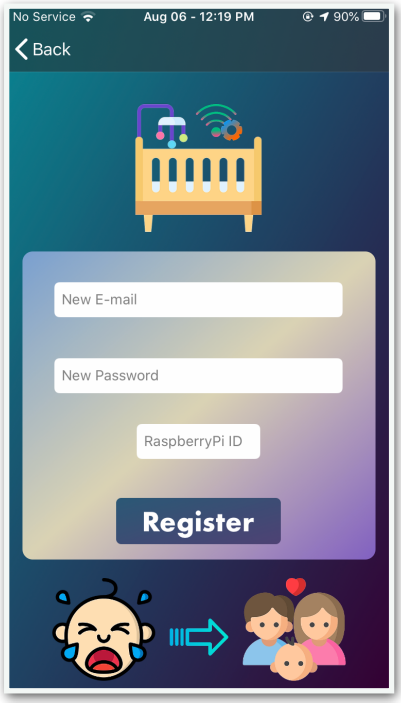
5.2.1 iOS Application

Login Screen

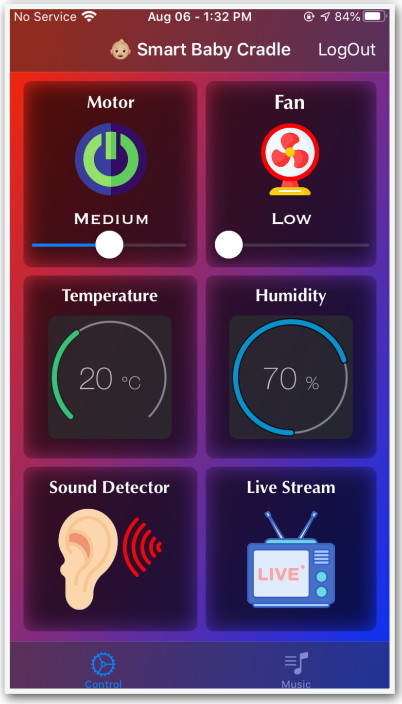


If you are new user

Register Screen



Navigate between Screens



Cradle Control Screen

Music Screen

Login Screen:

The Login screen to enter the email, the password and also check that user must be authorized, it also contains Sign Up button if the user doesn't have any account in our systems. ***In figure 5.11***

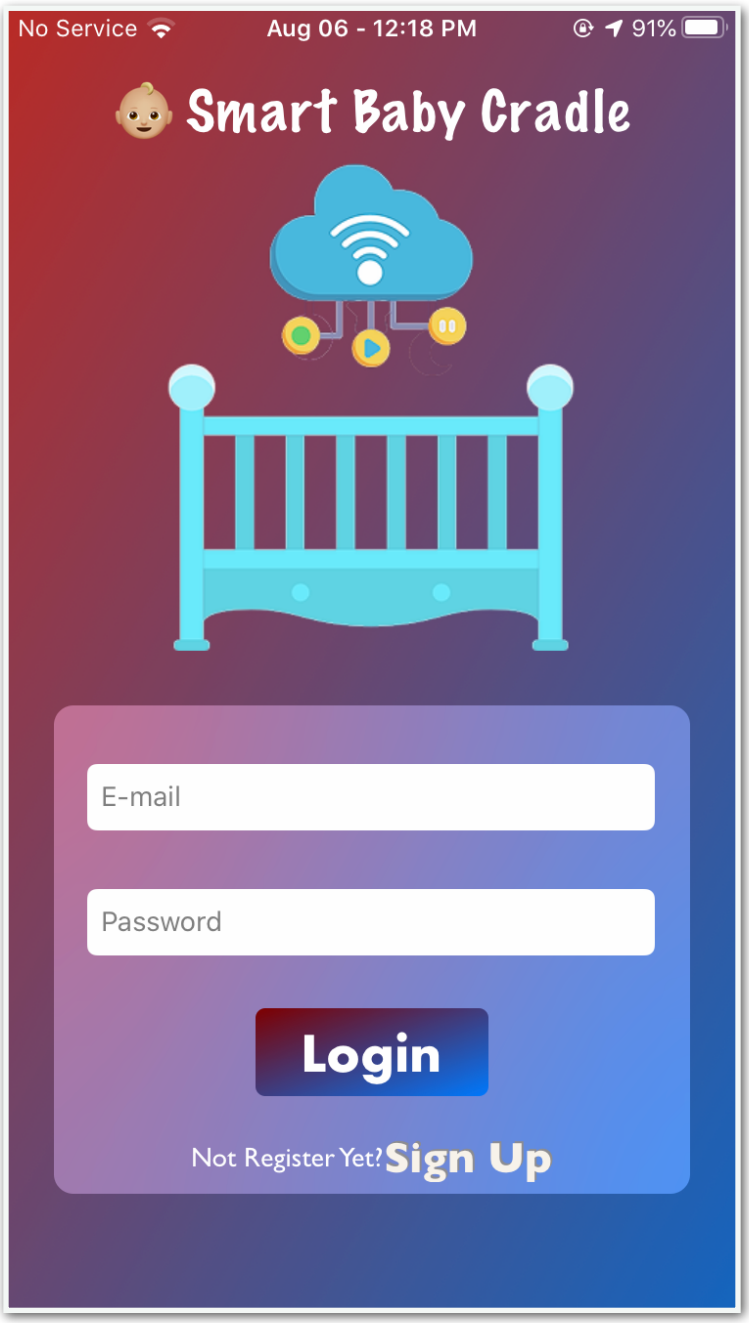


Figure 5.11: Login Screen

Sign Up Screen:

The Sign Up to register new user in our system and to check the raspberry pi ID device which in the baby cradle is the same ID will enter to avoid from anyone to register and control the application just for who buys the baby cradle.

In figure 5.12

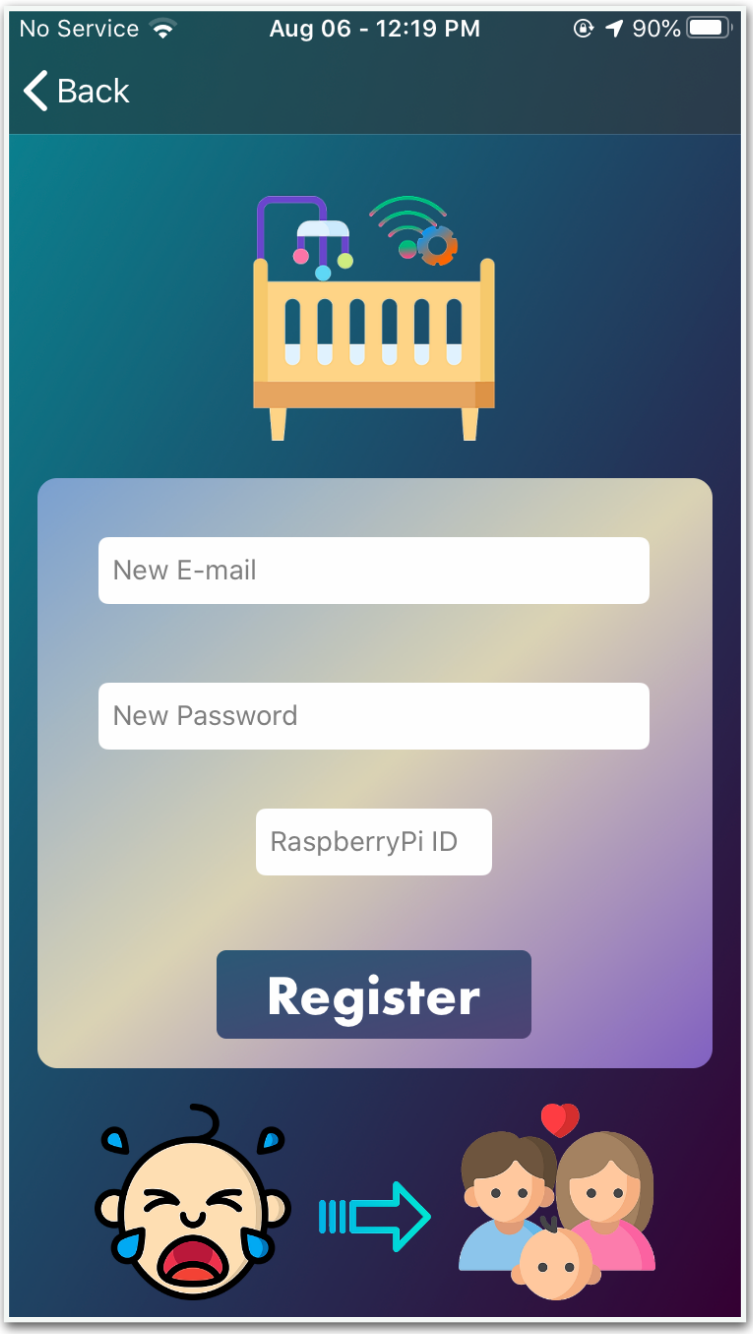


Figure 5.12: Sign Up Screen

Cradle Control Screen:

In that Screen will can you control the baby cradle by turning Motor On/Off and Fan On/Off with level degrees (Low-Medium-High), and also the parents will see the temperature of the baby, if baby urinates by humidity sensor , notify the parents if baby cries and watch the baby by live streaming from camera .

In figure 5.13

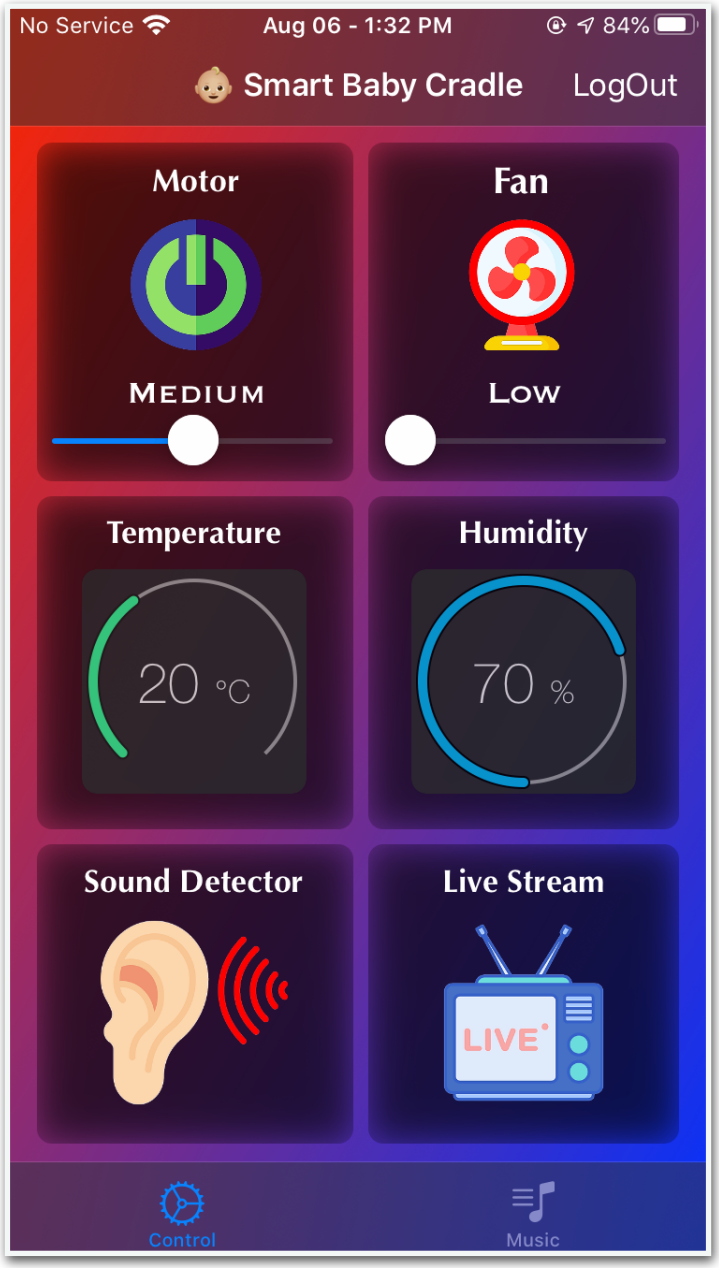


Figure 5.13: Cradle Control Screen

Music Screen :

In that screen will play song songs for baby if cries and want to calm down him or want to sleep. In Figure 5.14

Figure 5.14: Music Screen

Model-View-Controller (MVC) in iOS – Design Pattern :

It's Apple's recommended architecture pattern, and modify your code to be scalable and extensible.

As a new iOS developer, there is a huge amount of information you need to master: a new language, new frameworks and APIs, and Apple's recommended architectural pattern Model-View-Controller, or MVC for short.

The MVC pattern is commonplace in iOS development. While it's not a difficult pattern to understand and get familiar with, there are things to consider to avoid common pitfalls.

Getting up to speed with iOS development can be a daunting task and, more often than not, MVC doesn't get as much attention as the APIs or programming language. This can lead to major architectural problems in your apps down the road. ***In figure 5.15***

MVC is a software development pattern made up of three main objects:

- **The Model is where your data resides. Things like persistence, model objects, parsers, managers, and networking code live there.**
- **The View layer is the face of your app. Its classes are often reusable as they don't contain any domain-specific logic. For example, a UILabel is a view that presents text on the screen, and it is reusable and extensible.**
- **The Controller mediates between the view and the model via the delegation pattern. In an ideal scenario, the controller entity won't know the concrete view it's dealing with. Instead, it will communicate with an abstraction via a protocol. A classic example is the way a UIT able View communicates with its data source via the UIT able View Data Source protocol.**

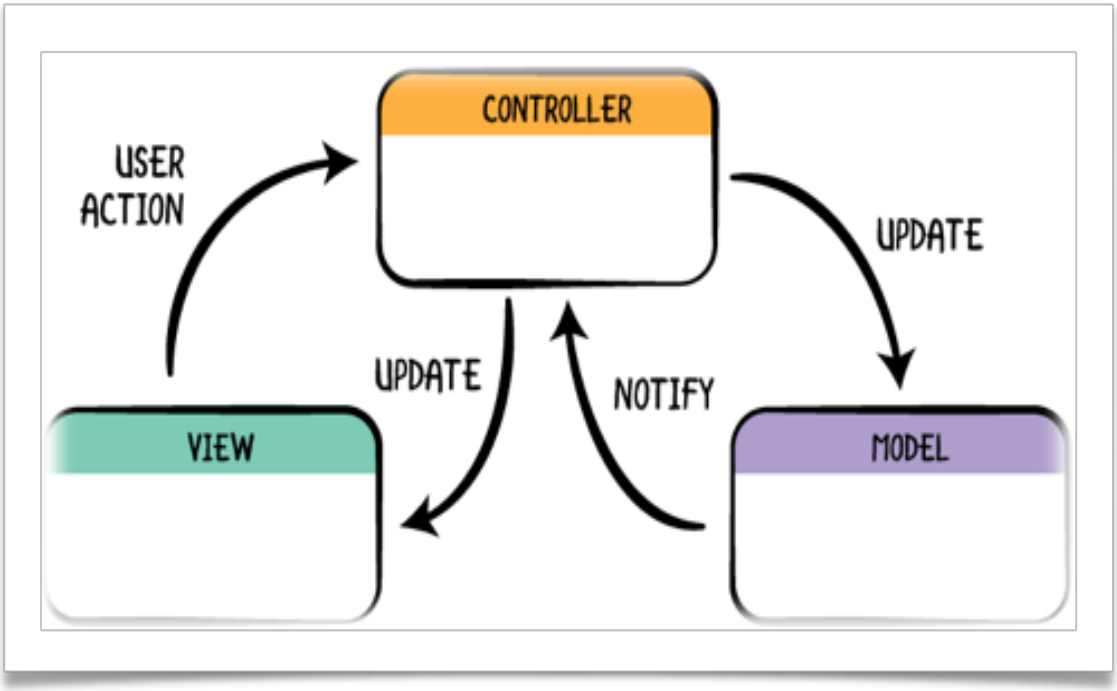


Figure 5.15: MVC Diagram

The Model (M):

The model layer encompasses your app's data. No surprise there, but there are usually other classes and objects in your projects that you can include in this layer:

- **Network code: You preferably only use a single class for network communication across your entire app. It makes it easy to abstract concepts common to all networking requests like HTTP headers, response and error-handling and more.**
- **Persistence code: You use this when persisting data to a database, Core Data or storing data on a device.**
- **Parsing code: You should include objects that parse network responses in the model layer. For example, in Swift model objects, you can use JSON encoding/decoding to handle parsing. This way, everything is self-contained and your network layer doesn't have to know the details of all your**

model objects in order to parse them. Business and parsing logic is all self-contained within the models.

- **Managers and abstraction layers/classes: Everyone uses them, everyone needs them, nobody knows what to call them or where they belong. It's normal to have the typical "manager" objects that often act as the glue between other classes. These can also be wrappers around lower-level, more robust API: a keychain wrapper on iOS, a class to help with notifications or a model to help you work with Health Kit.**
- **Data sources and delegates: Something that may be less common is developers relying on model objects to be the data source or delegate of other components such as table or collection views. It's common to see these implemented in the controller layer even when there's a lot of custom logic that's best kept compartmentalized.**
- **Constants: It's good practice to have a file with constants. Consider putting these within a structure of structures or variables. That way, you can reuse storyboard names, date formatters, colors, etc.**
- **Helpers and extensions: In your projects, you will often add methods to extend the capabilities of strings, collections, etc. These too can be considered part of the model.**

There are more classes and objects you could include, but these seem to be the most common. Again, the goal is not to focus too much on the semantics of what is or isn't a model. Rather, it is to have a solid foundation upon which to get your work done.

The View (V):

When users interacts with your app, they are interacting with the view layer. It should not contain any business logic. In code terms, you'll normally see:

UIView subclasses: These range from a basic UIView to complex custom UI controls.

- **Classes that are part of UIKit/AppKit.**
- **Core Animation.**
- **Core Graphics.**

Typical code smells found in this layer manifest in different ways but boil down to include anything unrelated to UI in your view layer. This can include network calls, business logic, manipulating models and so on.

Well-written view components are often reusable, but don't focus on that from the start. It's easy to get caught up in the thought of taking a cool button you built and making it compatible for a dozen different scenarios that your app may never need. Consider making components reusable only when you actually need to. When you have multiple use cases, that's the time to make a component more generic.

The Controller (C):

The controller layer is the least reusable part of your app as it often involves your domain-specific rules. It should be no surprise that what makes sense in your app won't always make sense in someone else's. The controller will then use all the elements in your model layer to define the flow of information in your app. Usually, you'll see classes from this layer deciding things like:
What should I access first: the persistence or the network?
How often should I refresh the app?
What should the next screen be and in which circumstances?
If the app goes to the background, what should I tidy up?
The user tapped on a cell; what should I do next?
Think of the controller layer as the brain, or engine, of the app; it decides what happens next.
You likely won't do much testing in the controller layer as, for the most part, it's meant to kick things off, trigger data loads, handle UI interactions, mediate between UI and model, etc.

Advantages:

- **Multiple developers can work simultaneously on the model, controller and views.**
- **Multiple developers can work simultaneously on the model, controller and views.**
- **MVC enables logical grouping of related actions on a controller together. The views for a specific model are also grouped together.**
- **Models can have multiple views.**

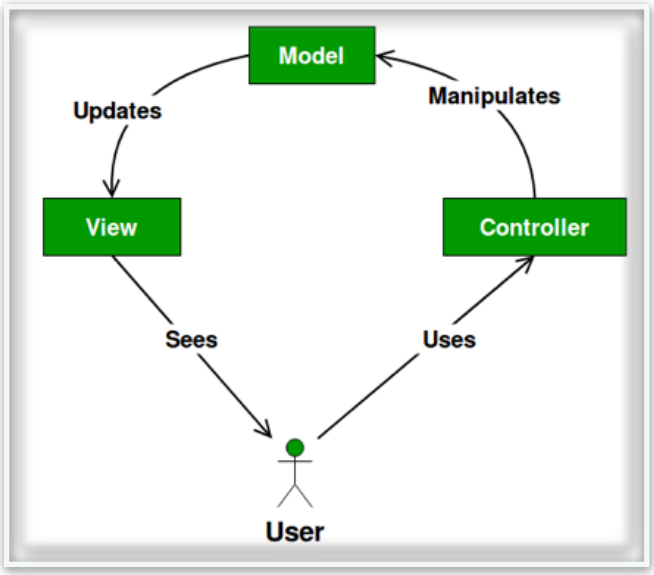


Figure 5.16: MVC Interactions with User

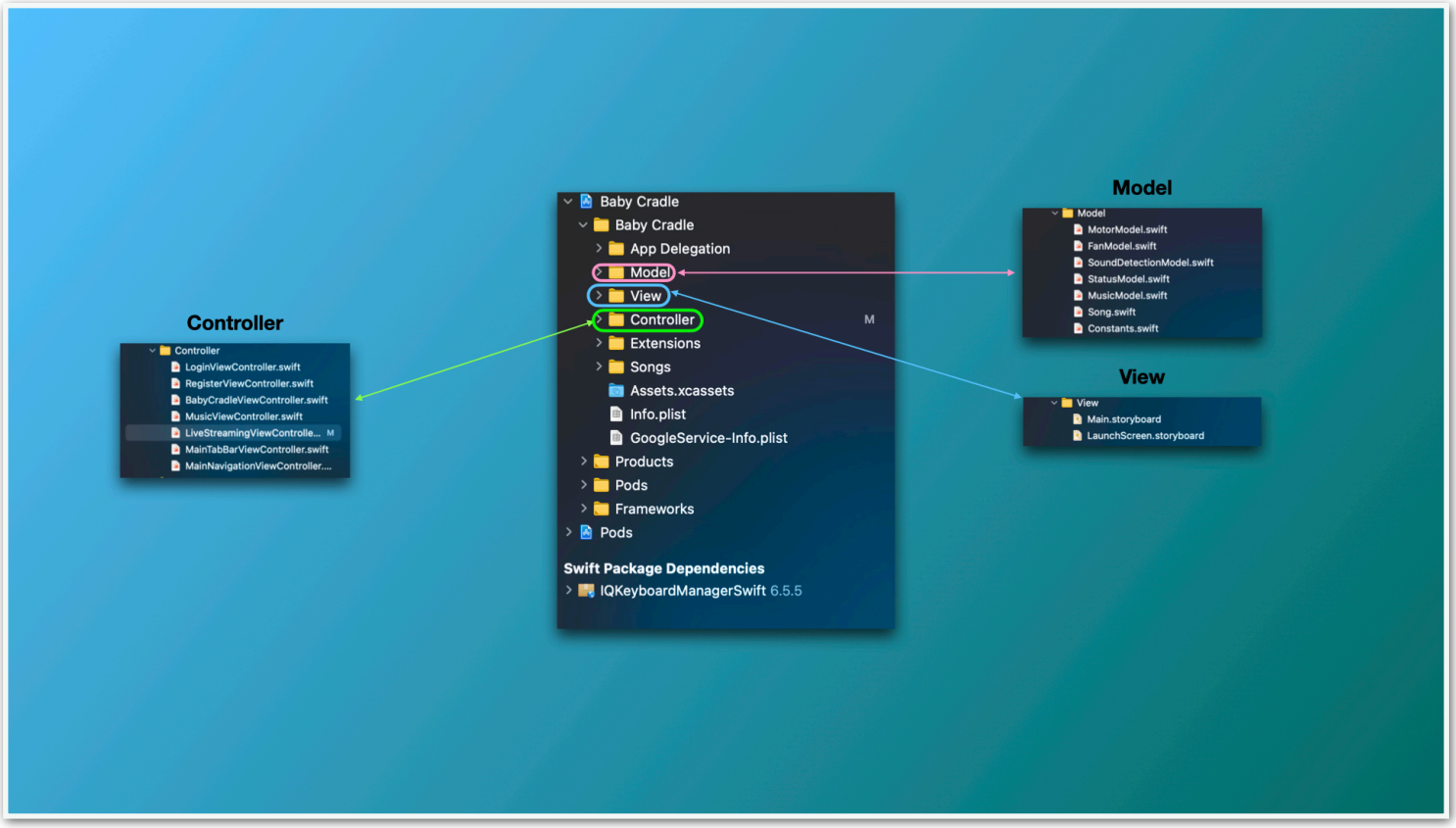
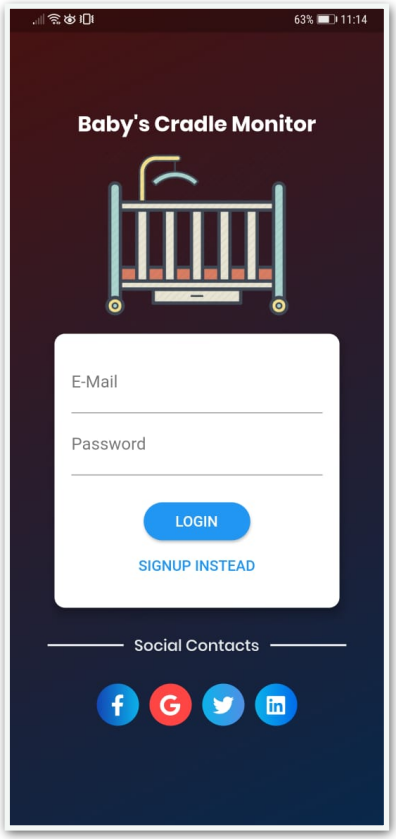


Figure 5.17: MVC in BMSCS iOS Project

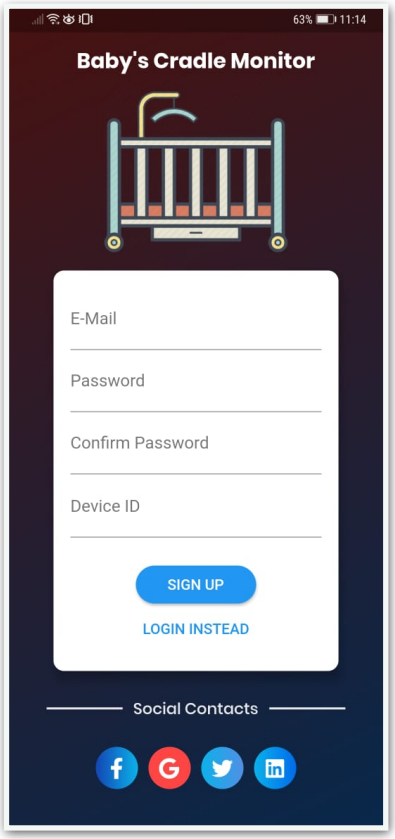
5.2.2 Android Application By Flutter

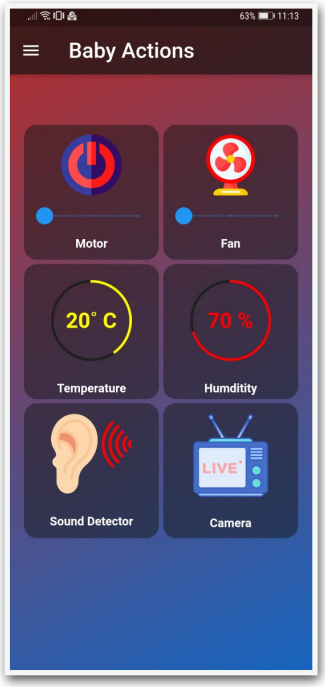
Login Screen



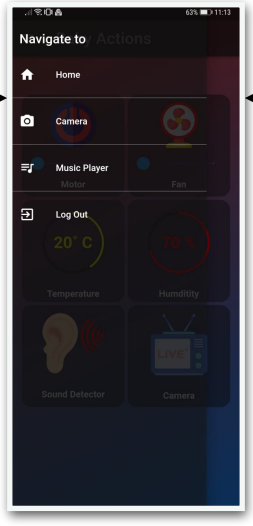
If You're New User

Sign Up Screen





Drawer



Cradle Control Screen

Music Screen

Authentication Screen :

After booting up, the app is wrapped in a wrapper widget this widget decides that the user is either authenticated or not and shows the appropriate screen.The default is trying to log the use in The other option is to sign up using a valid email, password and the device id. ***In figure (5.18, 5.19)***

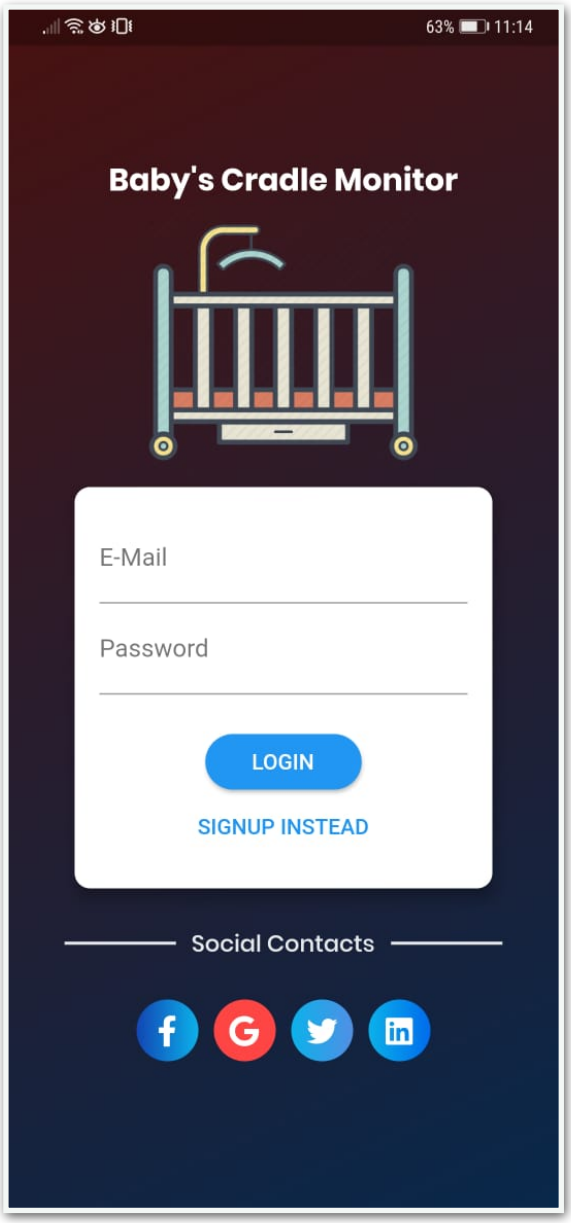


Figure 5.18: Login Screen

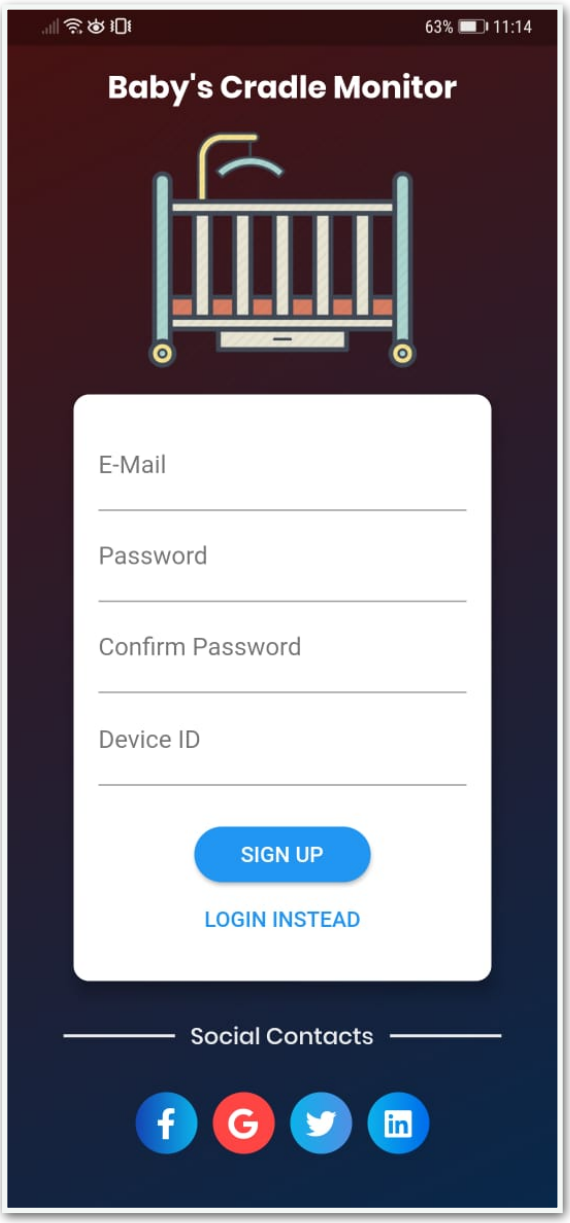


Figure 5.19: Figure Sign Up Screen

We are using firebase as our backend so we integrated error handling in our application.
Both in the log in and sign up firebase can detect any errors and send this error to our application. ***In figure (5.20, 5.21)***

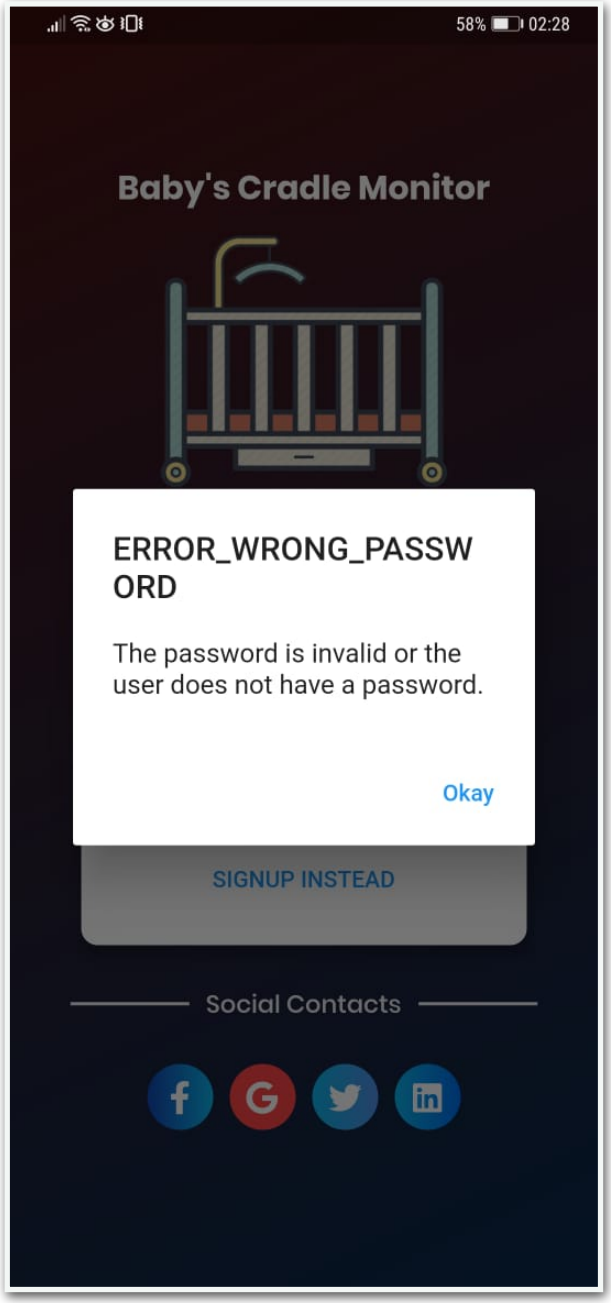


Figure 5.20: Invalid password

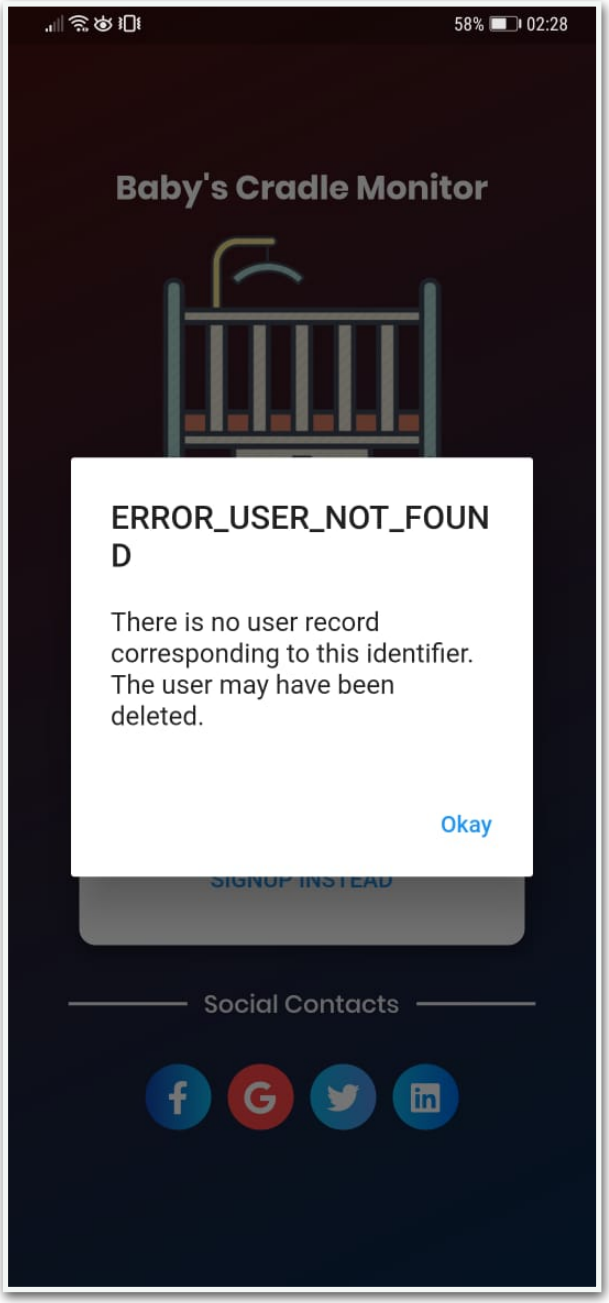


Figure 5.21: Invalid E-mail

Cradle Control Screen:

After a successful authentication or if the user is already logged in the app goes to the Cradle Control Screen. ***In figure 5.22,***This is the main screen of the app which includes the main operation:

- Controlling the Motor and Fan :

Switching them on or off and selecting an appropriate level. ***In figure 5.23***

- Getting the Temperature and Humidity :

We are using firebase ability to give us these reads and update them whenever they are changed in the database. ***In figure 5.24***

- Baby's Sound:

The baby's sound is monitored and when his crying is detected a notification sound is played and the icon is changed. ***In figure 5.25***

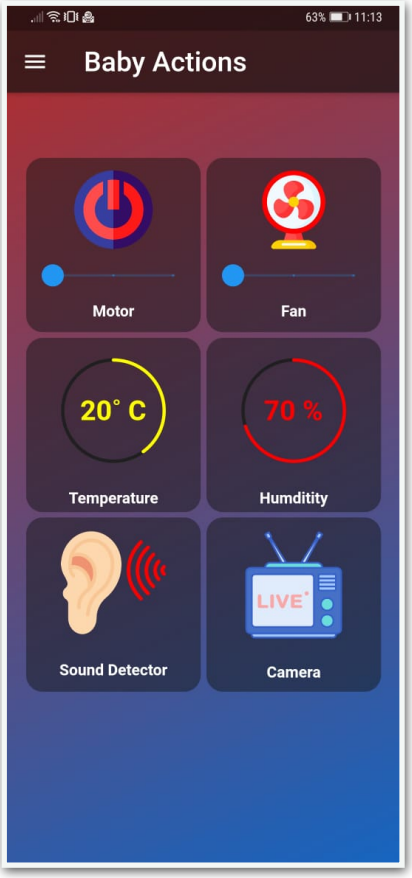


Figure 5.22: Cradle Control Screen

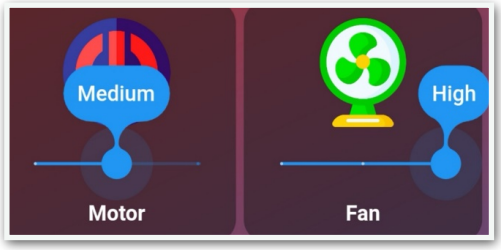


Figure 5.23: Motor & Fan Controller

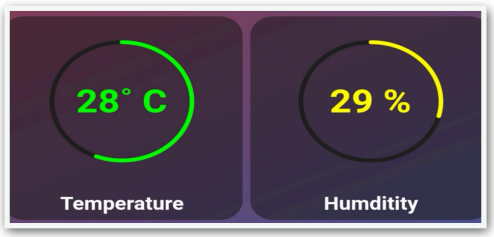


Figure 5.24: Temperature & Humidity Status

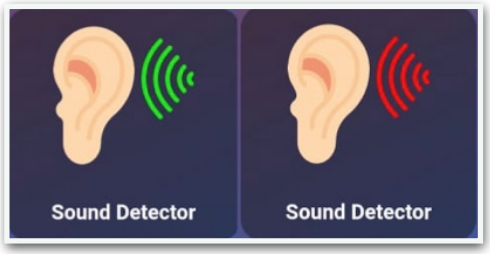


Figure 5.25: Sound detected

Music Screen :

It includes a basic media player selecting any songs plays it on the phone and on the baby's room . there is an option for a playlist and also muting the songs only on the phone. ***In figure 5.26***

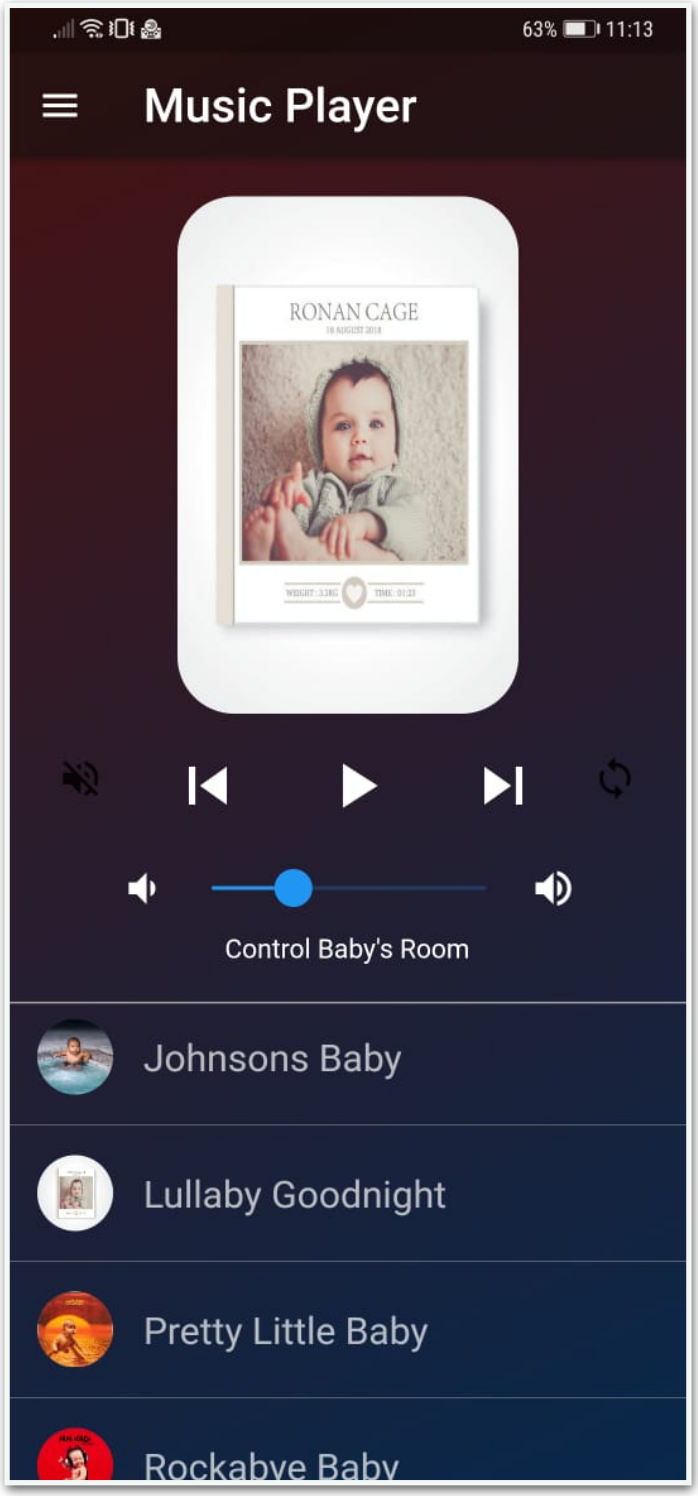


Figure 5.26: Music Screen

MVVM Design Pattern:

Flutter apps don't use any specific design pattern. This means the developer is in charge of choosing and implementing a pattern that fits their needs. The declarative nature of Flutter makes it an ideal candidate for the MVVM design pattern.

Understanding MVVM

MVVM stands for Model-View-View Model. The basic idea is to create a view model that'll provide data to the view. The view can use the data provided by the view model to populate itself. Creating a view-model layer allows you to write modular code, which can be used by several views.

This is extremely useful as we have multiple components that operates on the same logic but need different data so create one model which fetches data from firebase and sets a listener on this data allowing us to get any changes or updates and send them to our view to update the UI , This one model is then applied to these component with the small change of where to fetch the data from firebase. ***In figure 5.27***

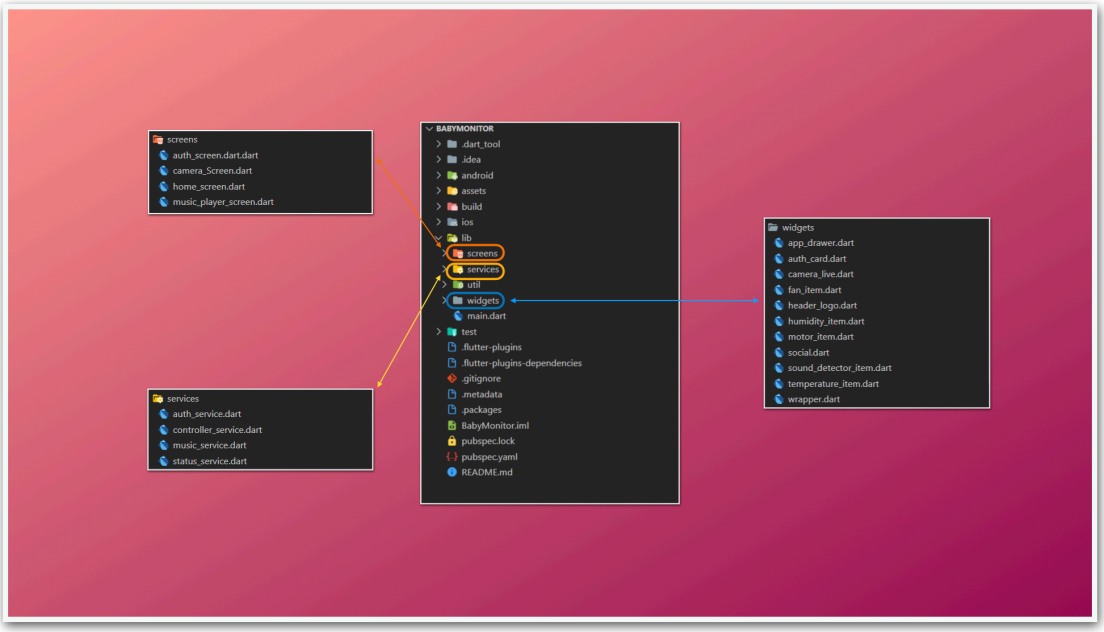


Figure 5.27: MVVM Design Pattern in Project Files

The first thing in our app:

1. Authentication:

As shown the app is wrapped in a widget that checks the authentication state of the user using the model in the right it uses the firebase SDK which provides a stream that monitor the auth state and the basic operations of signing up ,**In and Out**.

After the authentication check is done whether this is the first time the user is logging in or he has already logged in before the app goes to the Cradle Control screen and fetches data from firebase. ***In figure 5.28***

```
  Future signInWithEmailAndPassword(String email,
String password) async {
    try {
      AuthResult result = await _auth.
signInWithEmailAndPassword(
          email: email, password: password);
      FirebaseUser user = result.user;
      //return _userFormFirebase(user);
      return result;
    }
    catch (e){
      throw e;
    }

  }

  Future signOUt() async {
    try {
      return await _auth.signOut();
    } catch (e) {
      print(e.toString());
      return e;
    }
  }
}
```

Figure 5.28: Firebase Authentication SDK

2. Fetching data from the temperature and humidity :

This is the model used for both temperature and humidity , a request is sent to firebase using the method called get Status Once to get the stats then a listener is set on these values using get Status Stream it provides a stream that fires every time the data is changed so any changes done to them on the database is immediately sent to our app and updated in the UI. ***In figure(5.30, 5.31).*** The status model is imported to our view then a subscription is set to the stream of data coming from the firebase this allows our view to continuously update our UI every time a new value comes from firebase. ***In figure 5.29***

```
  Future<StreamSubscription<Event>> getStatusStream(
String directory , onData(Event event)) async {
    StreamSubscription<Event> subscription = database
        .reference()
        .child(directory)
        .onValue
        .listen((Event event) {
      onData(event);
    });

    return subscription;
  }

  Future<DataSnapshot> getStatusOnce(String
 directory) async {

    return database
        .reference()
        .child(directory)
        .once();

  }
}
```

Figure 5.29: Status Model

```
  @override
  void dispose() {
    if (_subscription != null) {
      _subscription.cancel();
    }
    super.dispose();
  }

  _updateTemp(Event event) {
    setState(() {
      _humditiyLevel = double.parse
(event.snapshot.value) / 100;
    });
  }
```

Figure 5.30: Fetching Humidity

```
@override
void dispose() {
  if (_subscription != null) {
    _subscription.cancel();
  }
  super.dispose();
}

_updateTemp(Event event) {
  setState(() {
    _temp = double.parse(event.snapshot.value);
  });
}
```

Figure 5.31: Fetching Temperature

Fetching Data from Sound Detector :

It's the same status model used in the temperature and humidity is also used here with the same principles.

The status model is imported to our view then a subscription is set to the stream of data coming from the firebase this allows our view to continuously update our UI every time a new value comes from firebase. ***in figure 5.32***

```
  @override
  void initState() {
    _soundStatus
        .getStatusStream(directory, _updateSound)
        .then((StreamSubscription
s) => _subscription = s);
    super.initState();
  }

  @override
  void dispose() {
    if (_subscription != null) {
      _subscription.cancel();
    }
    assetsAudioPlayer.dispose();
    super.dispose();
  }

  _updateSound(Event event) {
    setState(() {
      _sound = event.snapshot.value;
    });
    if (_sound == 'yes') {
      assetsAudioPlayer.open(Audio(
"assets/audios/Baby01.mp3"));
    }
  }
```

Figure 5.32: Fetching Data From Sound Detector

3. Fetching and updating data from/to the Motor and Fan Model:

This is the model used for controllers (motor and fan).

Like the temperature and humidity a request is sent to get the current state of the motor (the run and level), also a listener is set so if there are multiple users at the same time any changes done by one is reflected to the others this is done by setting up a subscription to the data stream in our view.

And we have the updateItem function that is used to update and control the firebase which is handled in the view according to user inputs. ***in figure 5.33***

```
  Future getStatusOnce(String directory) async {
    Completer completer = new Completer();

    FirebaseDatabase.instance
        .reference()
        .child(directory)
        .once()
        .then((DataSnapshot snapshot) {
      completer.complete(snapshot.value);
    });

    return completer.future;
  }


  Future<void> updateItem(String directory ,int
 status,double level)async{
     await database.reference().child(directory).
update({
      'level': level.toInt(),
      'run': status,
    });
  }


}
```

Figure 5.33: Motor & Fan Controller

4. **Music Service:**

The media player is always monitoring which song is playing, the sound level and updates the firebase accordingly using the update Firebase Song method & update Volume. ***In figure 5.34***

```
class _MusicPlayerScreenState extends State<
MusicPlayerScreen> {
  final musicService = MusicService();
  final AssetsAudioPlayer assetsAudioPlayer;
  _MusicPlayerScreenState(this
.assetsAudioPlayer);

  @override
  void initState() {
    assetsAudioPlayer.isPlaying.listen
((isPlaying) {
      assetsAudioPlayer.current.listen((songs) {
        changeImage(songs);
        musicService.updateFirebaseSong
(isPlaying, songs.index);
      });
    });
    musicService.updateFirebaseVolume(volume);

    super.initState();
  }
```

Figure 5.34: Music Service

5.2.2 Web Application

- **Authentication Screen**

(1) Register Page :

Allow users to fill form and register to app, if he new user. ***In figure 5.35.*** and saving it into database in firebase. ***In figure 5.36***

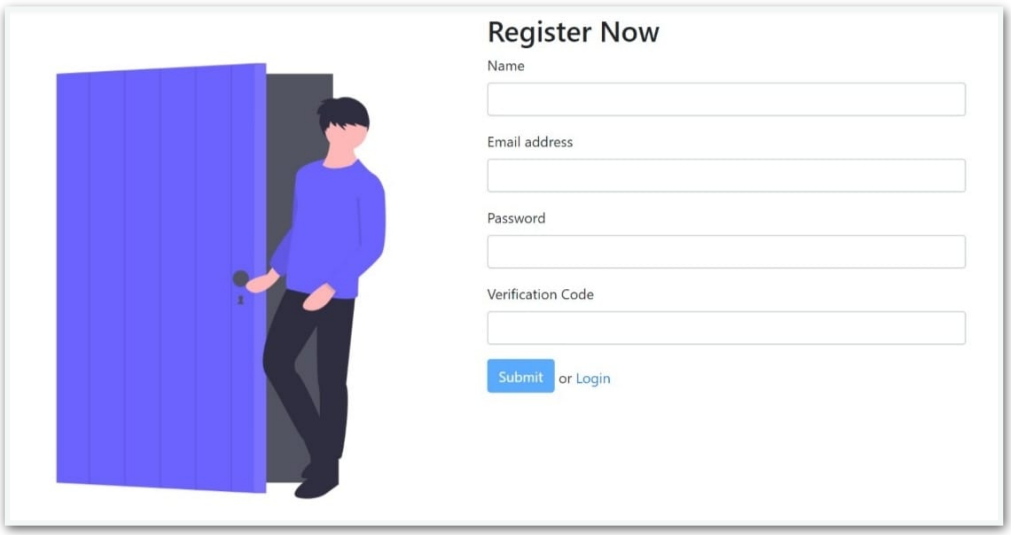


Figure 5.35: Register Page

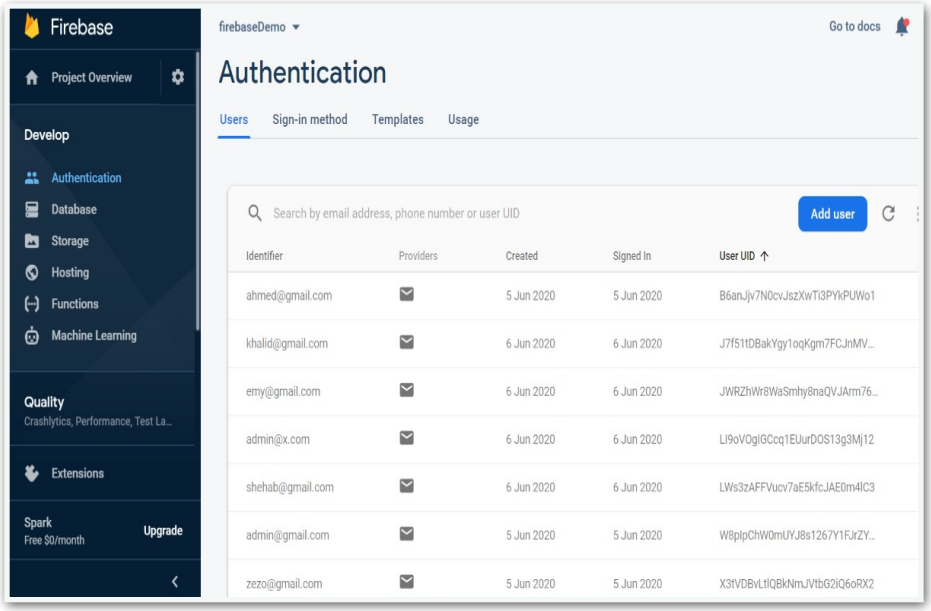


Figure 5.36: Authentication Databases in Firebase

(2) Login page :

To login user into app which fetching data of the user that exists in database in authentication of firebase. ***In figure 5.37***

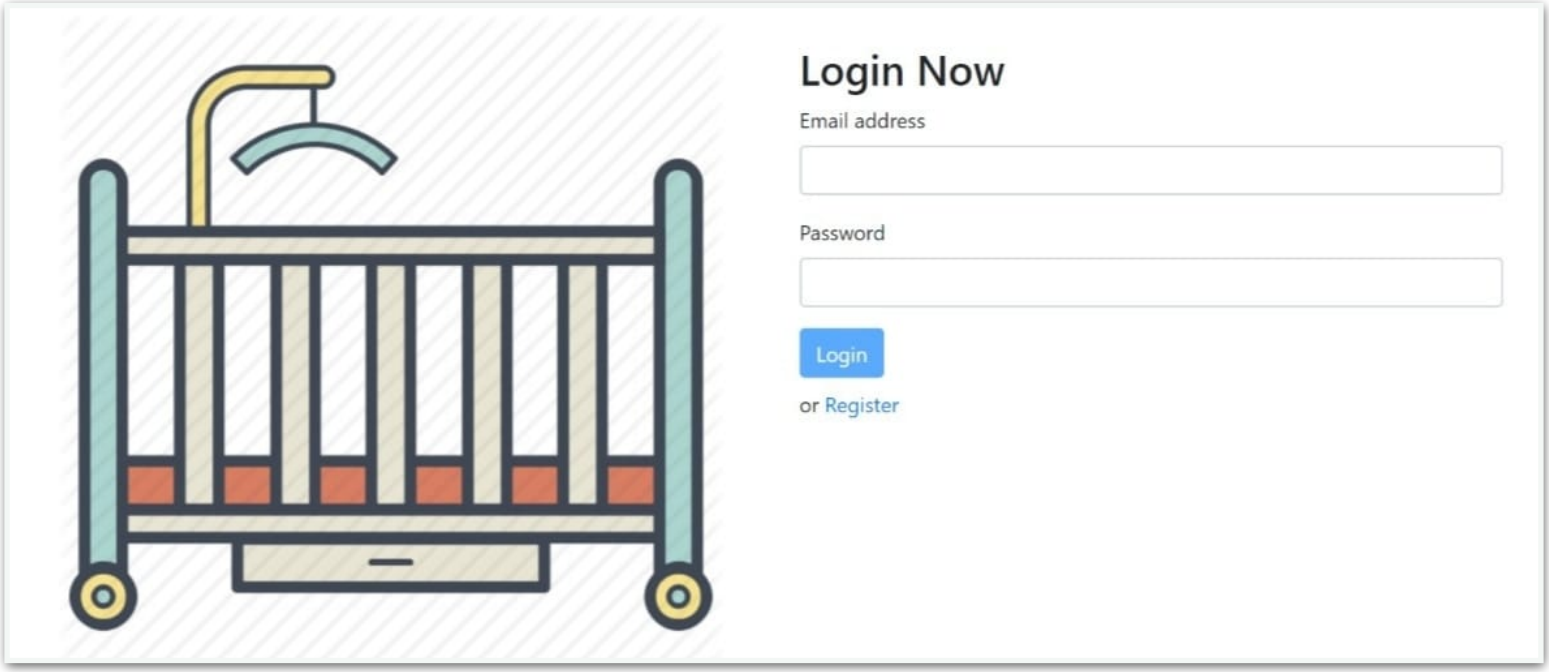


Figure 5.37: Login Page

▪ Cradle Control Screen

In that Screen will can you control the baby cradle by turning Motor On/Off and Fan On/Off with level degrees (Low-Medium-High), and also the parents will see the temperature of the baby, if baby urinates by humidity sensor , notify the parents if baby cries and watch the baby by live streaming from camera . ***In figure 5.38***

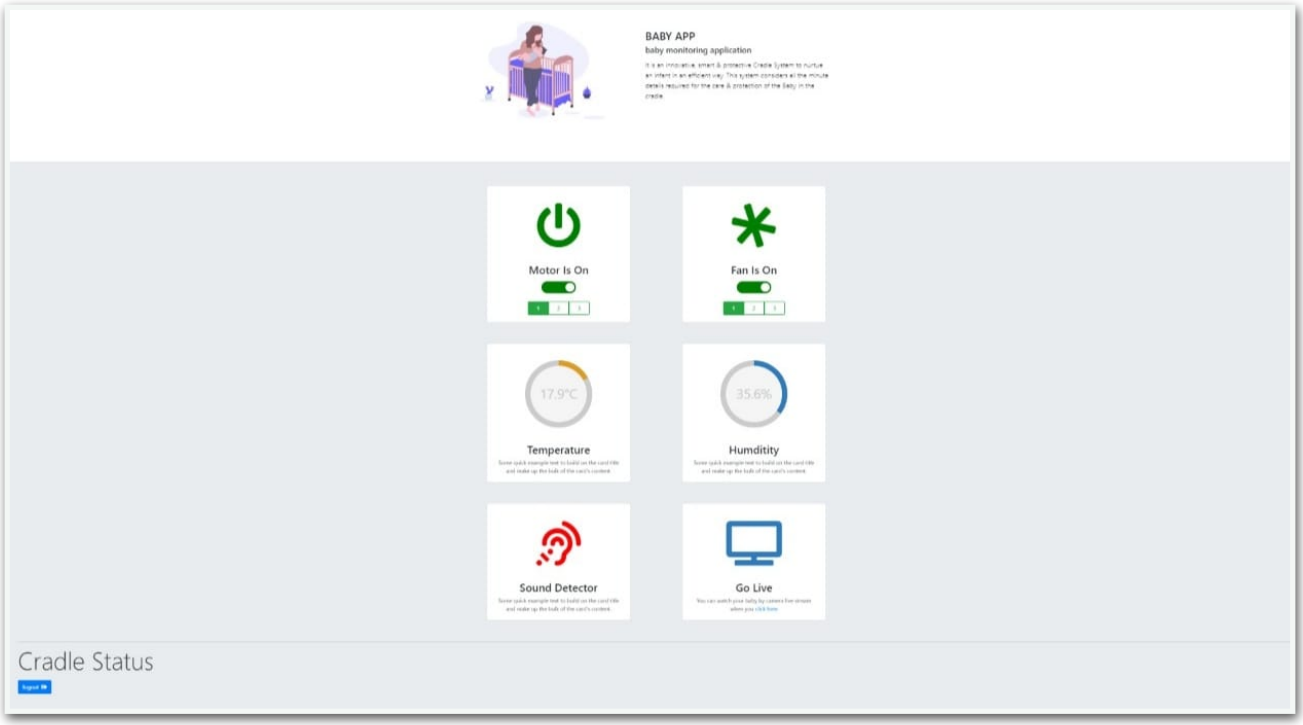


Figure 5.38: Cradle Control Screen

· Design Pattern

Angular framework is embedded with original **MVC** but it's more of an **MVVM** software architectural setup. ... Its framework uses the **MVVM**(**Model-View-ViewModel**) architecture better than an **MVC** (**Model-View-Controller**) one.
The **MVVM** model supports two-way data binding between View and ViewModel

• MVVM Design Pattern:

Let's begin by clarifying what MVVM is since it's far less ubiquitous than MVC. MVVM is a refinement of the MVC design whereby the ViewModel in MVVM facilitates the separation of development of the graphical user interface (UI) - i.e. the Presentation Layer.
The ViewModel (VM) is responsible for exposing (converting) the data objects from the model in such a way that objects are more easily managed and presented. In this respect, the ViewModel is more model than view, and handles most if not all of the view's display logic.

MVVM is a variation of Martin Fowler's Presentation Model design pattern in that a Presentation Model abstracts the View in a way that is not dependent on the specific UI platform. Hence, neither the Presenter or the View are aware of each other.
MVVM was invented by Microsoft architects Ken Cooper and Ted Peters in an effort to simplify event-driven programming of UIs. The pattern was incorporated into Windows Presentation Foundation (WPF) (Microsoft's .NET graphics system) and Silverlight (WPF's Internet application implementation).

John **Gossman**, one of **Microsoft's WPF** and Silverlight architects, first announced **MVVM** on his blog in 2005.
Model-view-view model is sometimes also referred to as model-view-binder to reflect the fact that the View handles behaviors, events and data binding information.
However, the **MVVM** format comes with its own flaws as well. Because it relies on data binding, the ViewModel consumes a considerable amount of memory in comparison to it's controlling counterparts. The creator of the **MVVM** pattern himself, John Gossman, said that the overhead for implementing **MVVM** is "overkill" for simple UI operations (Gossman). Larger applications that use the **ViewModel** method regularly become incredibly difficult to run. For this reason, the **MVVM** design pattern is used mostly for single page/function applications on the web. ***In figure 5.39***

Figure 5.39: MVVM Design Pattern

Chapter 6
6. Conclusion And Future Work

6.1 Hardware Improvement

---

1. Smart Camera for Raspberry Pi

We can support a smart camera that is capable of extracting application-specific information from the captured images, along with generating event descriptions or making decisions that are used in an intelligent and automated system. A smart camera is a self-contained, standalone vision system with a built-in image sensor in the housing of an industrial video camera, we take all of a camera's technical aspects into consideration such as video quality, audio, night vision, storage, and smart platform integrations. We also measure the overall value, convenience, and protection that these cameras offered during our testing.

=> Our preferences for security cameras are as follows:

- 1080p HD video. ***In figure 6.1***
- Two-way audio
- Optic zoom in-capability
- Field of view of at least 120 degrees
- Infrared night vision. ***In figure 6.4***
- Free cloud and local storage
- Person detection
- DIY installation or free professional installation
- User-friendly, companion mobile apps
- Thermal vision. ***In figure 6.3***

2. Intelligent Baby Behavior Monitoring

Target abnormal activity recognition by monitoring the baby to analyze baby behaviour, and to do that we need powerful camera with robust sensor vision. Also we can detect baby's expressions by using AI algorithms of face recognition, therefore We'll know that he's crying ,happy ,sad, hungry or sleeping ,etc…

3. Setting up database for Django Project

we can substitute firebase database with Django database

Django in its 'out-of-the-box' state is set up to communicate with SQLite -- a lightweight relational database included with the Python distribution. So, by default, Django automatically creates a SQLite database for your project.The Django configuration to connect to a database is done inside 'settting.py' file of a Django project in the DATABASES variable.

4. Setting up Camera Streaming

We can use Django to stream from camera but it needs a high performance raspberry pi.

5. Sending Notifications from advertising products to the Applications

We can use Django to make an API to search internet for products and offers that parents need or interested in so it can make it easy for the parents to find these products.

Ex: **milk**, **clothes**, **diapers**, etc…

6.2 Mobile Application

Software Development

It’s possible to add more functionality for the application.

- Make Drawer for Application and add some sections like:
  - Profile Picture for user
  - E-mail / Name
  - Credit
  - Help
  - Settings
- In Setting section when tap on it, we’ll go to its page where there:
  - Change (name/e-mail)
  - Parent gender
  - Age
  - Add user number
- Add reminder screen which has :
  - Sleeping time
  - Breast feeding time
  - Diaper changing time, etc..
- Add Status screen for baby which has:
  - Baby’s name
  - Baby’s age
  - Baby’s gender
  - Baby’s weight
- Add opening screen for application to inform user about application usage and what it's taking about.

Figure 6.1: Smart Camera

Figure 6.2

Figure 6.3: Thermal Vision

Figure 6.4: Night Vision

References

[1] Pi NoIR Camera V2 - Raspberry Pi. https://www.raspberrypi.org/ products/pi-noir-camera-v2/. Accessed 27 August 2019.

[2] Y. Lu and J. Cecil, ``An Internet of Things (IoT)-based collaborative framework for advanced manufacturing. '' *Int. J. Adv. Manuf. Technol.*, vol. 84, nos. 5_8, pp. 1141_1152, May 2016.

[3] R. Krishnamurthy and J. Cecil, ``A next-generation IoT-based collaborative framework for electronics assembly. '' *Int. J. Adv. Manuf. Technol.*, vol. 96,nos. 1_4, pp. 39_52, 2018.

[4] Read the UV4L documentation to find out more https://www.linux-projects.org/documentation/uv4l-server/

[5] Annouri J., Moyo T., Wang H., and Zuber D. Knights Wireless Baby Monitor 2014 Thesis at University of Central Florida.

[6] Pereira S. C., Shreelatha M. U., Nicole AT., and Pai S. Advanced Baby Monitor. International Journal of Internet of Things. 2017; 6(2):51-55.

[7] Aktas F., Kavus E., and Kavus Y. A Real-Time Infant Health Monitoring System for Hard of Hearing Parents by using Android-based Mobil Devices. Journal of Electrical and Electronics

[8] Reef-Nerd-Tutorial-Setting-up-a-Raspberry-Pi-Reef-Cam-and-Website

[9] Pygame Library: http://www.pygame.org/docs/ref/mixer.html?fbclid=IwAR2pKSSLHbtZLJ4fDptHbhGlsDDEbEjAeQw62QLHsb61uXTFdAnVJTzaXGo#pygame.mixer.get_busy

[10] DHT11 basic temperature-humidity sensor.
https://www.adafruit.com/product/386.\
Accessed 27 August 2019].

[11] Martinez G. and Lopez P. Advanced Databases Projects: Real-Time Databases and Firebase. Big Data Management and Analytics Master Program, University of Brussels, 2018.

[12] Caton G.. What's the safest temperature for my my baby's room?. https://www.babycentre.co.uk/
. Accessed 27 August 2019

[13] Apple :
https://developer.apple.com/documentation

Project Code

Baby Monitoring Smart Cradle System :

https://github.com/https-github-com-AbdelrahmanShehab