

1. INTRODUCTION

“SMART STICK” is an IOT based product which helps the visually impaired with the ability to commute to any place without any assistance. It can easily identify the obstacles in their vicinity using ultrasonic sensor, so that they can move accordingly. It enables them to easily switch between the modes of the stick depending on their requirement. Distance between the person and obstacle is calculated and information is stored in firebase (cloud database). Smart Stick android application retrieves the distance from firebase and audio feed is heard through speaker or earphone.

1.1 Motivation

People with complete blindness or low vision often have a difficult time self-navigating outside well-known environments. In fact, physical movement is one of the biggest challenges for blind people, explains World Access for the Blind. Travelling or simply walking down a crowded street may pose great difficulty. Because of this, many people with low vision will bring a sighted friend or family member to help navigate unknown environments. So, we decided to develop a product which helps them to self-navigate.

1.2 Problem Definition

Blind people must learn every detail about the home environment. Large obstacles such as tables and chairs must remain in one location to prevent injury. If a blind person lives with others, each member of the household must diligently keep walkways clear and all items in designated locations.

1.3 Objective of the Project

The objective of the project to track the distance of obstacles in the path of visually challenged person and the person should easily be able to know distance through audio format.

1.4 Limitations of the Project

The project is limited by the fact that it needs internet connectivity to send and retrieve the data.

2. Analysis

2.1 Introduction

“SMART STICK” is an IOT based product which helps the visually impaired with the ability to commute to any place without any assistance. It can easily identify the obstacles in their vicinity using ultrasonic sensor, so that they can move accordingly. It enables them to easily switch between the modes of the stick depending on their requirement. Distance between the person and obstacle is calculated and information is stored in firebase (cloud database). Smart Stick android application retrieves the distance from firebase and audio feed is heard through speaker or earphone.

2.2 Existing System

Visually impaired people have to use an ordinary cane or sighted person's assistance to walk. If they are alone, it might be quite difficult for them to perceive their surroundings and walk quickly. It becomes even more difficult for them to reach their destination safely if it is a crowded place.

2.3 Proposed System

The smart stick provides the visually impaired with the ability to commute to any place without any assistance. It can easily identify the obstacles in their vicinity so that they can move accordingly. It enables them to easily switch between the modes of the stick depending on their requirement. In manual mode, the user can press the speak button to know the distance between the obstacle and them. In the automatic mode, the smart stick repeatedly sends the obstacle distance in the form of speech at regular intervals of time.

2.4 Software Requirement Specification

2.4.1 Purpose

The main purpose of this project is to assist the visually impaired people in navigating their journey. In order to accomplish this, we make use of two ultrasonic sensors which are used to find the distance between the obstacle and the person. The distance obtained from the sensors is stored in the firebase. Then, the smart stick application retrieves the value of the distance and converts it into text-to-speech for the user to hear.

2.4.2 Scope

This Product can be used in different scenarios like

- Can be used to navigate through crowded areas.
- It helps the visually impaired to navigate through the path in night time.
- It allows the visually impaired to travel on their own.

2.4.3 Overall Description

The Smart Stick Device makes use of Ultrasonic Sensors that detects objects by sending a short ultrasonic burst and then listening for the echo. The Raspberry Pi connected to the sensors calculates the distance from the object based on the time the echo took to come back. Next, we send the distance that is calculated to the Firebase. Simultaneously, in the android application, the value of the distance is retrieved from the firebase and a text output is generated based on the distance calculated. This Text output is converted into an Audio Format, which is then heard using an Earphone or a Speaker of visually impaired person's smartphone. This system could be integrated on a stick, making it portable.

2.4.4 External Interface Requirements

Since the distance obtained from the raspberry pi have to send the data to the firebase server, the whole device and the mobile phone must be connected to internet over wireless connection.

2.4.5 Technologies Used

Software Requirements

- **Python**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL).Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and UNIX shell and other scripting languages. Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in

directing its progress. Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by meta-programming and meta-objects. Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has `filter()`, `map()`, and `reduce()` functions; list comprehensions, dictionaries, and sets; and generator expressions. The standard library has two modules that implement functional tools borrowed from Haskell and Standard ML.

- **Raspbian OS**

Raspbian is a Debian-based computer operating system for Raspberry Pi. There are several versions of Raspbian including Raspbian Stretch and Raspbian Jessie. Since 2015 it has been officially provided by the Raspberry Pi Foundation as the primary operating system for the family of Raspberry Pi single-board computers. Raspbian was created by Mike Thompson and Peter Green as an independent project. The initial build was completed in June 2012. The operating system is still under active development. Raspbian is highly optimized for the Raspberry Pi line's low-performance ARM CPUs.

Raspbian uses PIXEL, **Pi Improved Xwindows Environment**, **Lightweight** as its main desktop environment as of the latest update. It is composed of a modified LXDE desktop environment and the Openbox stacking window manager with a new theme and few other changes. The distribution is shipped with a copy of computer algebra program Mathematica and a version of Minecrafti called Minecraft Pias well as a lightweight version of Chromium as of the latest version. Raspbian is an unofficial port of Debian Wheezy armhf with compilation settings adjusted to produce optimized "hard float" code that will run on the Raspberry Pi. This provides significantly faster performance for applications that make heavy use of floating point arithmetic operations. All other applications will also gain some performance through the use of advanced instructions of the ARMv6 CPU in

Raspberry Pi. Although Raspbian is primarily the efforts of Mike Thompson (mptompson) and Peter Green (plugwash), it has also benefited greatly from the enthusiastic support of Raspberry Pi community members who wish to get the maximum performance from their device.

- **PuTTY**

PuTTY is an open-source application making use of network protocols like Telnet and rlogin in Windows and UNIX platforms in conjunction with an xterm terminal emulator. Over a network, PuTTY makes use of all the above protocols to enable a remote session on a computer. It is a popular tool for text-based communication and is also a popular utility for connecting Linux servers from Microsoft operating system-based computers.

PuTTY is a free SSH, Telnet and Rlogin client for Windows systems. SSH, Telnet and Rlogin are three ways of doing the same thing: logging in to a multi-user computer from another computer, over a network. Multi-user operating systems, such as UNIX and VMS, usually present a command-line interface to the user, much like the ‘Command Prompt’ or ‘MS-DOS Prompt’ in Windows. The system prints a prompt, and you type commands which the system will obey.

Using this type of interface, there is no need for you to be sitting at the same machine you are typing commands to. The commands, and responses, can be sent over a network, so you can sit at one computer and give commands to another one, or even to more than one.

SSH, Telnet and Rlogin are *network protocols* that allow you to do this. On the computer you sit at, you run a *client*, which makes a network connection to the other computer (the *server*). The network connection carries your keystrokes and commands from the client to the server, and carries the server's responses back to you.

These protocols can also be used for other types of keyboard-based interactive session. In particular, there are a lot of bulletin boards, talker systems and MUDs (Multi-User Dungeons) which support access using Telnet. There are even a few that support SSH.

The primary goal of PuTTY is to become a multi-platform application capable of executing in most operating systems. It can be considered like an xterm terminal for most

purposes. It even specifies its terminal type as xterm to the server; although this can be reconfigured. Most features like port forwarding and public keys are available through the command line options.

The main window of PuTTY has the session which runs on the remote computer and through which one can send the commands directly to the remote computer. For reducing the unpredictability of random data, PuTTY makes use of a random number seed file which is usually stored in PUTTY.RND file. With regards to cut and paste features, PuTTY can be customized to act similarly to xterm.

PuTTY provides some distinct advantages, especially when working remotely. It is easier to configure and is more stable. It is also more persistent in comparison to others, as a remote session can be resumed as soon the connection is restored after an interruption.

It has an easy-to-use graphical user interface. Many variations on the secure remote terminal are supported by PuTTY. Some terminal control sequences like the Linux console sequences which are unsupported by xterm are supported by PuTTY.

- **Firebase**

The Firebase Real-time Database provides an application with a cloud-hosted NoSQL database. Data is synced across connected clients as soon as data is changed. Firebase also supports offline mode, writing database changes to a local database before sending them off to the Firebase server for synchronization with other devices.

Firebase also provides client SDKs for web, Android, and iOS as well as a REST API for integrations with your own server. Firebase real-time database Stores and sync data with firebase NoSQL cloud database. Data is synced across all clients in real-time, and remains available when application goes offline.

The Firebase Real-time Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client. When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.

Firebase offers two cloud-based, client-accessible database solutions that support real-time data syncing, they are realtime Database and cloud firestore. Realtime Database is

Firebase's original database. It's an efficient, low-latency solution for mobile apps that require synced states across clients in realtime. Cloud Firestore is Firebase's new flagship database for mobile app development.

It improves on the successes of the Realtime Database with a new, more intuitive data model. Cloud Firestore also features richer, faster queries and scales better than the Realtime Database.

With just a single API, the Firebase database provides your app with both the current value of the data and any updates to that data. Realtime syncing makes it easy for your users to access their data from any device, be it web or mobile. Realtime Database also helps your users collaborate with one another.

Another benefit of Realtime Database is that it ships with mobile and web SDKs, allowing you to build your apps without the need for servers. When your users go offline, the Realtime Database SDKs use local cache on the device to serve and store changes. When the device comes online, the local data is automatically synchronized.

The Realtime Database can also integrate with Firebase Authentication to provide a simple and intuitive authentication process.

- **Android**

Android is a mobile operating system initially developed by Android Inc. Android was purchased by Google in 2005. Android is based upon a modified version of the Linux kernel. Google and other members of the Open Handset Alliance collaborated to develop and release Android to the world.

The Android Open Source Project (AOSP) is tasked with the maintenance and further development of Android. Unit sales for Android OS Smartphone ranked first among all Smartphone OS handsets sold in the U.S. in the second and third quarters of 2010, with a third quarter market share of 43.6%.

Android has a large community of developers writing application programs ("apps") that extend the functionality of the devices. There are currently over 100,000 apps available

for Android. Android Market is the online app store run by Google, though apps can be downloaded from third party sites (except on AT&T, which disallows this). Developers write in the Java language, controlling the device via Google-developed Java libraries.

The unveiling of the Android distribution on 5 November 2007 was announced with the founding of the Open Handset Alliance, a consortium of 79 hardware, software, and telecom companies devoted to advancing open standards for mobile devices. Google released most of the Android code under the Apache License, a free software and open source license.

The Android operating system software stack consists of Java applications running on a Java based object oriented application framework on top of Java core libraries running on a Dalvik virtual machine featuring JIT compilation.

Libraries written in C include the surface manager, Open Coremedia framework, SQLite relational database management system, OpenGL ES 2.0 3D graphics API, Web Kit layout engine, SGL graphics engine, SSL, and Bionic libc. The Android operating system consists of 12 million lines of code including 3 million lines of XML, 2.8 million lines of C, 2.1 million lines of Java, and 1.75 million lines of C++.

The fact that hitherto dedicated devices such as mobile phones can now count themselves among the venerable general-computing platforms is great news for programmers. This new trend makes mobile devices accessible through general-purpose computing languages, which increases the range and share for mobile applications.

The Android Platform embraces the idea of general-purpose computing for handheld devices. It is a comprehensive platform that features a Linux-based operating system stack for managing devices, memory, and processes. Android's libraries cover telephony, video, graphics, UI programming, and a number of other aspects of the device.

The Android SDK supports most of the Java Platform, Standard Edition (Java SE) except for the Abstract Window Toolkit (AWT) and Swing. In place of AWT and Swing, Android SDK has its own extensive modern UI framework. Because you're programming your applications in Java, you could expect that you need a Java Virtual Machine (JVM) that is responsible for interpreting the runtime Java byte code.

A JVM typically provides the necessary optimization to help Java reach performance levels comparable to compiled languages such as C and C++. Android offers its own optimized JVM to run the compiled Java class files in order to counter the handheld device

limitations such as memory, processor speed, and power. This virtual machine is called the Dalvik virtual machine.

The familiarity and simplicity of the Java programming language coupled with Android's extensive class library makes Android a compelling platform to write programs for. Let us look at how Android arrived on the Mobile OS landscape. Mobile phones use a variety of operating systems such as Symbian OS, Microsoft's Windows Mobile, Mobile Linux, iPhone operating system (based on Mac OS X), Moblin (from Intel), and many other proprietary operating systems.

So far, no single operating system has become the de facto standard. The available APIs and environments for developing mobile applications are too restrictive and seem to fall behind when compared to desktop frameworks. This is where Google comes in. The Android platform promised openness, affordability, open source code, and a high-end development framework.

Google acquired the start-up company Android Inc. in 2005 to start the development of the Android Platform. The key players at Android Inc. included Andy Rubin, Rich Miner, Nick Sears, and Chris White. In late 2007, a group of industry leaders came together around the Android Platform to form the Open Handset Alliance. Some of the alliance's prominent members are as follows:

1. Sprint Nextel
2. T-Mobile
3. Motorola
4. Samsung
5. Sony Ericsson
6. Toshiba
7. Vodafone
8. Google
9. Intel
10. Texas Instruments

Part of the alliance's goal is to innovate rapidly and respond better to consumer needs, and its first key outcome was the Android Platform. Android was designed to serve the needs of mobile operators, handset manufacturers, and application developers. The members have committed to release intellectual property through the open source Apache License.

The Android SDK was first issued as an “early look” release in November 2007. In September 2008, T-Mobile announced the availability of T-Mobile G1, the first Smartphone based on the Android platform. A few days after that, Google announced the availability of Android SDK Release Candidate 1.0. In October 2008, Google made the source code of the Android platform available under Apache’s open source license.

When Android was released, one of its key architectural goals was to allow applications to interact with one another and reuse components from one another. This reuse not only applies to services, but also to data and the user interface (UI). As a result, the Android platform has a number of architectural features that keep this openness a reality.

Android has also attracted an early following because of its fully developed features to exploit the cloud-computing model offered by web resources and to enhance that experience with local data stores on the handset itself. Android’s support for a relational database on the handset also played a part in early adoption.

In late 2008 Google released a handheld device called Android Dev Phone 1 that was capable of running Android applications without being tied to any cell phone provider network. The goal of this device (at an approximate cost of \$400.00) was to allow developers to experiment with a real device that could run the Android OS without any contracts.

At around the same time, Google also released a bug fix, version 1.1 of the OS that is solely based on version 1.0. In releases 1.0 and 1.1 Android did not support soft keyboards, requiring the devices to carry physical keys. Android fixed this issue by releasing the 1.5 SDK in April 2009, along with a number of other features, such as advanced media-recording capabilities, widgets, and live folders.

Hardware Requirements

- **Raspberry Pi 1 Model B+**

The Raspberry pi is a single computer board with credit card size that can be used for many tasks that your computer does, like games, word processing, spreadsheets and also to play HD video. It was established by the Raspberry pi foundation from the UK. It has been ready for public consumption since 2012 with the idea of making a low-cost educational microcomputer for students and children. The main purpose of designing the raspberry pi board is, to encourage learning, experimentation and innovation for school level students.

The raspberry pi board is a portable and low cost. Maximum of the raspberry pi computers is used in mobile phones. In the 21st century, the growth of mobile computing technologies is very high, a huge segment of this being driven by the mobile industries. The 98% of the mobile phones were using ARM technology.

The printed circuit board (PCB) houses the input and output connectors as well as the computer hardware itself. Currently, the Foundation is selling a naked PCB — meaning there is no included Raspberry Pi case and will release a cheaper version, with fewer connectivity options, soon.

These two versions without cases are essentially the beta run and buildup to the release of the final product. The final version will be an educational edition with case, documentation, and pre-loaded educational software. On the software side of things, there are currently three Linux-based operating systems supported by the Raspberry Pi.

The raspberry pi comes in two models, they are model A and model B. The main difference between model A and model B is USB port. Model A board will consume less power and that does not include an Ethernet port.

But, the model B board includes an Ethernet port and designed in china. The raspberry pi comes with a set of open source technologies, i.e. communication and multimedia web technologies. In the year 2014, the foundation of the raspberry pi board launched the computer module that packages a model B raspberry pi board into module for use as a part of embedded systems, to encourage their use.

The raspberry pi board comprises a program memory (RAM), processor and graphics chip, CPU, GPU, Ethernet port, GPIO pins, Xbee socket, UART, power source connector. And various interfaces for other external devices. It also requires mass storage, for that we use an SD flash memory card. So that raspberry pi board will boot from this SD card similarly as a PC boots up into windows from its hard disk.

The Raspberry Pi is open hardware, with the exception of the primary chip on the Raspberry Pi, the Broadcom SoC (System on a Chip), which runs many of the main components of the board—CPU, graphics, memory, the USB controller, etc. Many of the projects made with a Raspberry Pi are open and well-documented as well and are things you can build and modify yourself.

Essential hardware specifications of raspberry pi board mainly include SD card containing Linux OS, US keyboard, monitor, power supply and video cable. Optional hardware specifications include USB mouse, powered USB hub, case, internet connection, the Model A or B: USB WiFi adaptor is used and internet connection to Model B is LAN cable.

The Raspberry Pi was designed for the Linux operating system, and many Linux distributions now have a version optimized for the Raspberry Pi. Two of the most popular options are Raspbian, which is based on the Debian operating system, and Pidora, which is based on the Fedora operating system.

For beginners, either of these two work well; which one you choose to use is a matter of personal preference. A good practice might be to go with the one which most closely resembles an operating system you're familiar with, in either a desktop or server environment.

If you would like to experiment with multiple Linux distributions and aren't sure which one you want, or you just want an easier experience in case something goes wrong, try NOOBS, which stands for New Out Of Box Software. When you first boot from the SD card, you will be given a menu with multiple distributions (including Raspbian and Pidora) to choose from.

If you decide to try a different one, or if something goes wrong with your system, you simply hold the Shift key at boot to return to this menu and start over. There are, of course, lots of other choices. OpenELEC and RaspBMC are both operating system distributions based on Linux that are targeted towards using the Raspberry Pi as a media center.

There are also non-Linux systems, like RISC OS, which run on the Pi. Some enthusiasts have even used the Raspberry Pi to learn about operating systems by designing their own.

- **2x Ultrasonic sensor – HC-SR04**

An Ultrasonic sensor is a device that can measure the distance to an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back. By recording the elapsed time between the

sound wave being generated and the sound wave bouncing back, it is possible to calculate the distance between the sonar sensor and the object.

Since it is known that sound travels through air at about 344 m/s (1129 ft/s), you can take the time for the sound wave to return and multiply it by 344 meters (or 1129 feet) to find the total round-trip distance of the sound wave.

Round-trip means that the sound wave travelled 2 times the distance to the object before it was detected by the sensor; it includes the 'trip' from the sonar sensor to the object AND the 'trip' from the object to the Ultrasonic sensor (after the sound wave bounced off the object). To find the distance to the object, simply divide the round-trip distance in half.

The accuracy of Ultrasonic sensor can be affected by the temperature and humidity of the air it is being used in. However, for these tutorials and almost any project you will be using these sensors in, this change in accuracy will be negligible.

An optical sensor has a transmitter and receiver, whereas an ultrasonic sensor uses a single ultrasonic element for both emission and reception. In a reflective model ultrasonic sensor, a single oscillator emits and receives ultrasonic waves alternately. This enables miniaturization of the sensor head.

It is important to understand that some objects might not be detected by ultrasonic sensors. This is because some objects are shaped or positioned in such a way that the sound wave bounces off the object, but are deflected away from the Ultrasonic sensor.

It is also possible for the object to be too small to reflect enough of the sound wave back to the sensor to be detected. Other objects can absorb the sound wave all together (cloth, carpeting, etc.), which means that there is no way for the sensor to detect them accurately. These are important factors to consider when designing and programming a robot using an ultrasonic sensor.

- **SD card manager**

SD cards, short for Secure Digital, are everywhere you look now, from digital cameras, to phones and tablets, and even Single Board Computers (SBCs). In many cases your SBC won't come with Linux or any other operating system on it. It is up to you to provide the OS on an SD card. With the exception of Noobs for Raspberry Pi, this is usually not a drag and drop procedure.

Before Raspberry pi is powered up, SD card should be programmed with an Operating System. Much like a computer having Windows, Mac OS X or Linux on it to make it run, the Raspberry Pi needs something to help it boot and run software.

The card class determines the sustained write for the card; a class 4 card will be able to write at 4MB/s, whereas a class 10 should be able to attain 10MB/s. However, it should be noted that this does not mean a class 10 card will outperform a class 4 card for general usage, because often this write speed is achieved at the cost of read speed and increased seek times.

The original Raspberry Pi Model A and Raspberry Pi Model B require full-size SD cards. The newer Raspberry Pi Model A+, Raspberry Pi Model B+, Raspberry Pi 2 Model B, Raspberry Pi Zero, and Raspberry Pi 3 Model B require micro SD cards.

- **Android Smart Phone**

Android is a smartphone operating system (OS) developed by Google. It is used by a variety of mobile phone manufacturers including Motorola, HTC and Sony Ericsson.

The first phone to run on the Android OS was the HTC Dream which was launched on 22 October 2008. Google's own-brand Android phone, the HTC-made Nexus One, was launched in the US on 5 January 2010. Each version features slightly different functionality and user interface, and recently-launched software may or may not work on older versions of the OS. The Android cell phone is a cell phone running the Android OS. A typical Android cell phone is a smartphone with a touch screen interface, multiple connectivity options, Internet browsing capabilities, support for video playback and a camera.

Android is an open-source operating system which means that any manufacturer can use it in their phones free of charge. The term Android cell phone is actually a generic term. Unlike iPhone phones or BlackBerry phones, which always refer to phones manufactured by Apple and RIM (Research In Motion) respectively, Android phones may refer to a wide selection of phones from different manufacturers, including Motorola, HTC, Lenovo, LG, Samsung and Sony Ericsson. Some of the Android cell phone models gaining a lot of public attention include Motorola Droid X, HTC Dream, Google Nexus One and Samsung Galaxy. Android sales have been showing the fastest growth among smart phones since the first Android phone was released in 2008. The first Android phone was the HTC Dream, also called the T-Mobile G1.

- **Regulated +5V power supply**

Raspberry pi requires a power source of 5V to be operational and we have to insert a Micro SD memory card in it, which acts as its permanent memory. We can use a battery, portable charger, micro USB or a rectifier as the input power source.

3. Design

3.1 Project Architecture

Software Development Life Cycle (SDLC)

SDLC Methodologies

This document play a vital role in the Software Development Life Cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

Spiral Model

Spiral Model was defined by Barry Boehm in his 1988 article, “A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development. The spiral model is similar to the incremental model, with more emphasis placed on risk analysis.

The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model).

The baseline spirals, starting in the planning phase, requirements are gathered and risk is assessed. Each subsequent spiral builds on the baseline spiral. As originally envisioned, the iterations were typically 6 months to 2 years long.

Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

The steps for Spiral Model can be generalized as follows:

- The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
 - A preliminary design is created for the new system.
 - A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
- A second prototype is evolved by a fourfold procedure:
 1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
 2. Defining the requirements of the second prototype.
 3. Planning and designing the second prototype.
 4. Constructing and testing the second prototype.
- At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involve development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product.
- The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
- The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
- The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

- **Planning Phase**

Requirements are gathered during the planning phase. Requirements like 'BRS' that is 'Business Requirement Specifications' and 'SRS' that is 'System Requirement specifications'. The Project Planning Phase is the second phase in the project life cycle. It involves creating of a set of plans to help guide your team through the execution and closure phases of the project.

The plans created during this phase will help you to manage time, cost, quality, change, risk and issues. They will also help you manage staff and external suppliers, to ensure that you deliver the project on time and within budget. There are 10 Project Planning steps you need to take to complete the Project Planning Phase efficiently.

The Project Planning Phase is often the most challenging phase for a Project Manager, as you need to make an educated guess of the staff, resources and equipment needed to complete your project. You may also need to plan your communications and procurement activities, as well as contract any 3rd party suppliers.

In short, you need to create a comprehensive suite of project plans which set out a clear project roadmap ahead. The Planning Phase begins when the project has been formally approved and funded, and the Project Charter is approved.

This Phase requires study and analysis culminating in the full Project Management Plan and that may lead to system development activities. Acquisition activities are performed, if necessary, to obtain contractor support. The project work is broken down into specific tasks and sub-tasks, including the identification of project deliverables and assignment of allocated resources to each task. Control documents relating to that effort are produced.

The degree of project management rigor that is to be applied to the project is determined and milestones are established. Specific plans for management and governance of the project are established and documented to guide ongoing project execution and control.

The Planning Phase ends with a formal review during which the adequacy of the Project Management Plan is determined. In the planning phase, sufficient requirements detail is

required to support the development of the project's Project Management Plan and permit outside validation of this deliverable.

The Project Manager works with the CIO and Business Owner to verify the scope of the proposed program, participation of the key organizations, and potential individuals who can participate in the formal reviews of the project.

This decision addresses both programmatic and information management-oriented participation as well as technical interests in the project that are known at this time. The PM plans the subsequent phases to allow development of the project schedule and budget requirements, and to define the expected performance benefits. The PM also prepares a Project Process Agreement that specifies project deliverables and their expected levels of detail, and documents the justification for tailoring EPLC elements, if any.

Detailed activities and timelines for preparing acquisition documents, selecting vendors, and awarding contracts are developed. The Integrated Project Team identifies all alternatives that may address the need and any programmatic or technical risks.

The risks associated with further development are also studied. The results of these assessments are summarized in the Business Case and the Project Management Plan. To ensure that Privacy Act considerations are addressed early in the project lifecycle, the PM also prepares a Privacy Impact Assessment.

- **Risk Analysis**

Risk management is a process that provides management with the balance of meeting business objectives or missions and the need to protect the assets of the organization cost effectively.

In this period of increased external scrutiny due to the myriad questionable management decisions and the corresponding legislative backlash, risk management provides management with the ability to demonstrate actively due diligence and how they are meeting their fiduciary duty.

Once a project has been approved, early in the next phase of the BPDC, the design phase, a risk assessment must be performed to identify the threats presented by this new project to the organization's mission or business objectives.

The risk assessment allows the development team and the business stakeholders to identify potential threats, prioritize those threats into risks, and identify controls that can reduce the risks to acceptable levels.

Knowing the control requirements in the design phase will help reduce costs when work begins on the project in the construction or development phase. Risk analysis is the process that allows management to demonstrate that it has met its obligation of due diligence when making a decision about moving forward with a new project, capital expenditure, investment strategy, or other such business process.

Due diligence has a number of variant definitions based on the industry that is being discussed. Typically, the consensus these definitions address is the measure of prudent activity, or assessment, as is properly to be expected from, and ordinarily exercised by, a reasonable and prudent person under the particular circumstances.

Due diligence is not measured by any absolute standard but depends on the relative facts of each case. In brief, the risk analysis or PIA examines the factors that come into play when trying to determine if a project should be approved.

The PIA examines the tangible impacts; e.g., capital outlay, development costs, and long-term costs such as continued operations and maintenance. The risk analysis also addresses intangible impacts, such as customer connivance or regulatory compliance.

When the risk analysis is complete, the results are presented to a management oversight committee that is charged with reviewing new project requests and deciding whether or not to move forward.

If the request is approved, the project is registered and a risk assessment is scheduled for early in the design phase of the BPDC or SDLC. The documentation is retained for a period of time and then can be used by the organization if ever there are any questions about why a project was or was not approved. The output from the risk analysis and risk assessment processes will generally be used twice.

The first time will be when decisions are made: for the risk analysis, that means deciding whether or not to proceed on a new project; for the risk assessment, that means identifying the types of controls or safeguards that need to be implemented.

For risk assessment, the output will identify what countermeasures should be implemented or document that management has determined that the best decision is to accept the risk. The other time the results will be used is when the "spam hits the fan."

That is, when a problem arises and the organization must show the process it used to reach the decisions that it did. The documentation created in the risk management processes will allow the organization to show who was involved, what was discussed, what was considered, and what decisions were made.

By implementing risk analysis and risk assessment, an organization has the tools in place to make informed business decisions. By integrating these processes across the entire enterprise, the organization can take back control of its activities from outside interference.

With an effective risk assessment process in place, only those controls and safeguards that are actually needed will be implemented. An enterprise will never again face having to implement a mandated control to "be in compliance with audit requirements."

- The following diagram shows how a spiral model acts like

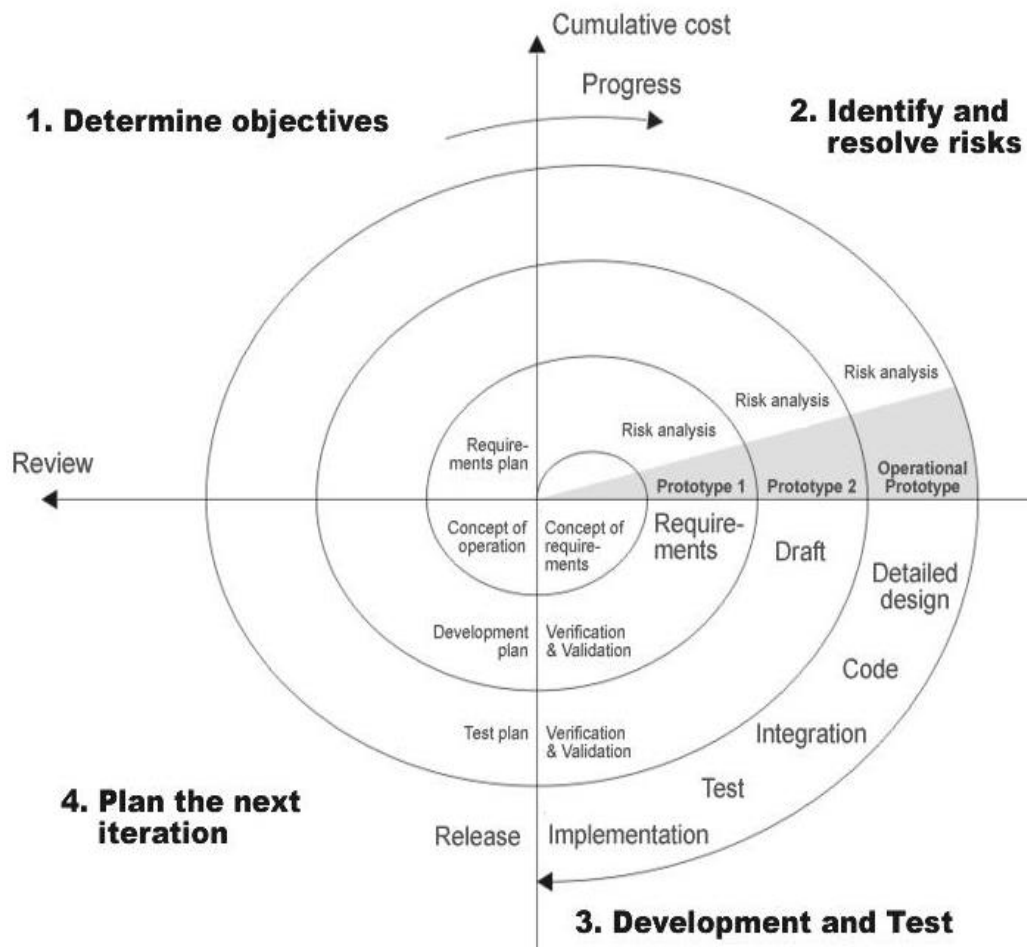


Figure.3.1.1: Spiral Model

- **Engineering Phase**

In this phase software is developed, along with testing at the end of the phase. Hence in this phase the development and testing is done. After the requirements and design activity is completed, the next phase of the Software Development Life Cycle is the implementation or development of the software. In this phase, developers start coding according to the requirements and the design discussed in previous phases.

Database admins create the necessary data in the database, front-end developers create the necessary interfaces and GUI to interact with the back-end all based on guidelines and procedures defined by the company.

Developers also write unit tests for each component to test the new code that they have written, review each other's code, create builds and deploy software to an environment. This cycle of development is repeated until the requirements are met. In this phase software is developed, along with testing at the end of the phase. Hence in this phase the development and testing is done. During testing, experienced testers start to test the system against the requirements.

The testers aim to find defects within the system as well as verifying whether the application behaves as expected and according to what was documented in the requirements analysis phase. Testers can either use a test script to execute each test and verify the results, or use exploratory testing which is more of an experience based approach. It is possible that defects are identified in the testing phase.

Once a defect is found, testers inform the developers about the details of the issue and if it is a valid defect, developers will fix and create a new version of the software which needs to be verified again. This cycle is repeated until all requirements have been tested and all the defects have been fixed and the software is ready to be shipped.

- **Evaluation phase**

This phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral. The testers aim to find defects within the system as well as verifying whether the application behaves as expected and according to what was documented in the requirements analysis phase.

Testers can either use a test script to execute each test and verify the results, or use exploratory testing which is more of an experience based approach. It is possible that defects are identified in the testing phase.

Once a defect is found, testers inform the developers about the details of the issue and if it is a valid defect, developers will fix and create a new version of the software which needs to be verified again.

This cycle is repeated until all requirements have been tested and all the defects have been fixed and the software is ready to be shipped.

3.2 Design Modules

Input Design

Considering the requirements, procedures to collect the necessary input data in most efficiently designed. The input design has been done keeping in view that, the interaction of the user with the system being the most effective and simplified way.

Also the measures are taken for the following:

- Controlling the amount of input.
- Eliminating extra steps.
- Keeping the process simple.
- At this stage the input forms and screens are designed.

Output Design

All the systems in the screen are designed with the view to provide the user with easy operations in simpler and efficient way, minimum key strokes possible. Almost every screen is provided with no error and option selection facilitates.

Emphasis is given for speedy processing and speedy transaction between the screens. Each screen assigned to make it as much user friendly as possible by using interactive procedures. So to say a user can operate the system without any ambiguity.

3.3 Architecture

The main component of our project is the raspberry pi 1. We connect two ultrasonic sensors to raspberry pi. Ultrasonic sensors are used for obstacle detection and calculation of distance between the obstacle and the visually impaired person. Ultrasonic sensors are used in pair as *transceivers* i.e. a single sensor can both send and receive signals.

The transmitter emits eight 40 kHz pulse, this pulse after hitting the obstacle is received back at the receiver. The ultrasonic sensor works on the principle of *sonar* i.e. it records the time taken by the emitted pulse to return back at the receiver end. Our algorithm implemented in Python programming language is deployed on Raspberry Pi.

This algorithm is used to calculate the distance between the obstacle and the person, by recording the time interval between the pulse sent and pulse received. After the distance is successfully calculated, the value of the distance is sent to the firebase real-time database. This is accomplished using the “firebase.put()” method.

Now, the database will consist of the shortest distance obtained from the ultrasonic sensors. Finally, the android application retrieves the value of the distance from the firebase through the internet. It then converts the text to speech using TTS module. This results in the user receiving a response in the form of a speech.

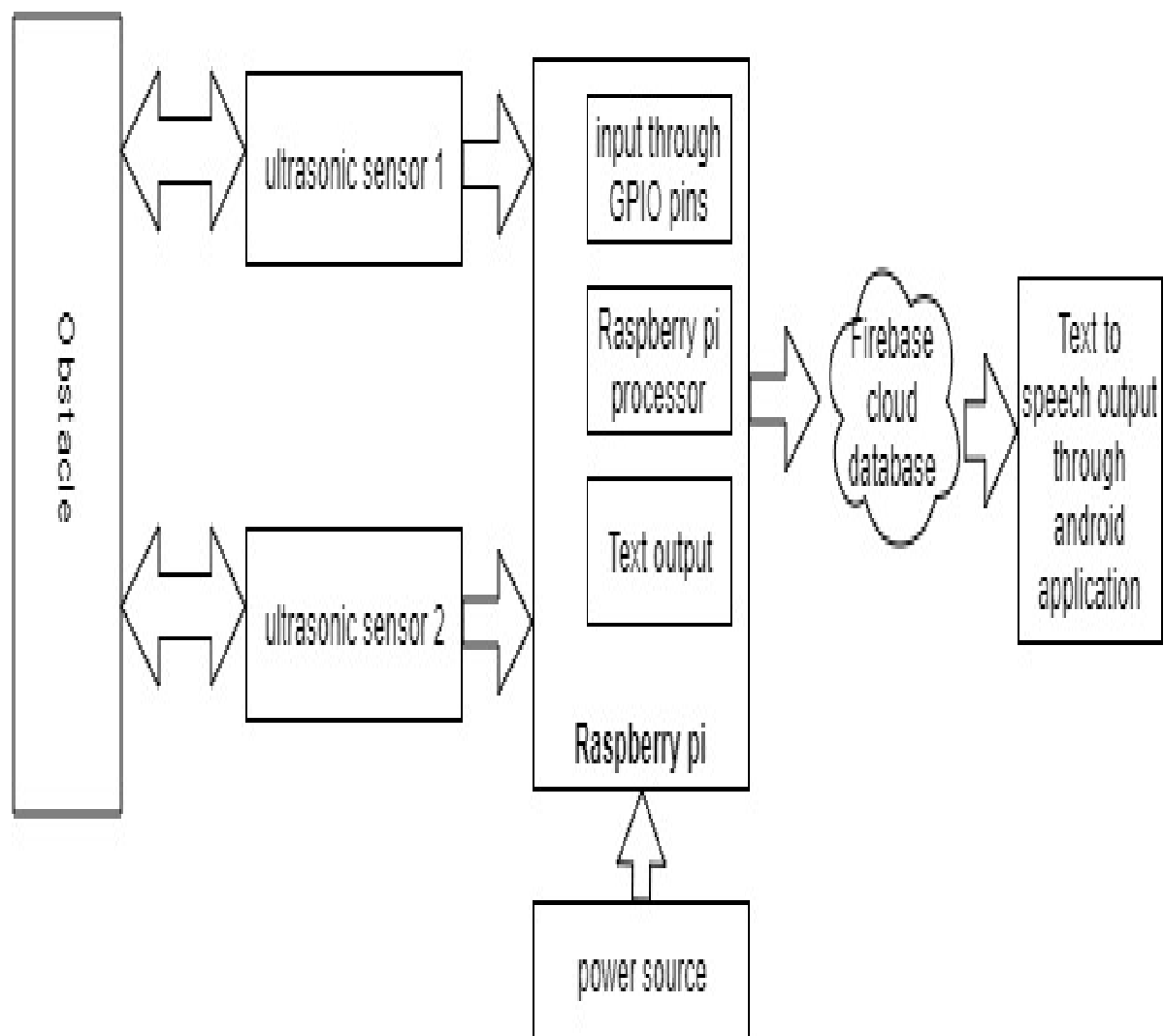


Figure.3.3.1 Architecture of Smart Stick

A. Raspberry pi:

Raspberry Pi is a credit card sized single board, low cost computer. It takes input from the GPIO pins, which can be attached to LEDs, switches, analogous signals and other devices. For our proposed design, we connect the GPIO pins to the ultrasonic sensors. It requires a power source of 5V to be operational and we have to insert a Micro SD memory card in it, which acts as its permanent memory.

For our design Raspberry Pi 1 Model B+ is used. It contains 4 USB ports, a HDMI port, an audio jack port and an Ethernet port. The Ethernet port helps the device connect to the Internet and install required driver APIs. It has a 700 MHz single core processor and supports programming languages such as Python, Java, C, and C++ etc.

This minicomputer runs our algorithm, which helps to calculate the distance from the obstacle based on the input it receives from the sensors. This system requires a 5V power supply. We can use a battery, portable charger, micro USB or a rectifier as the input power source.

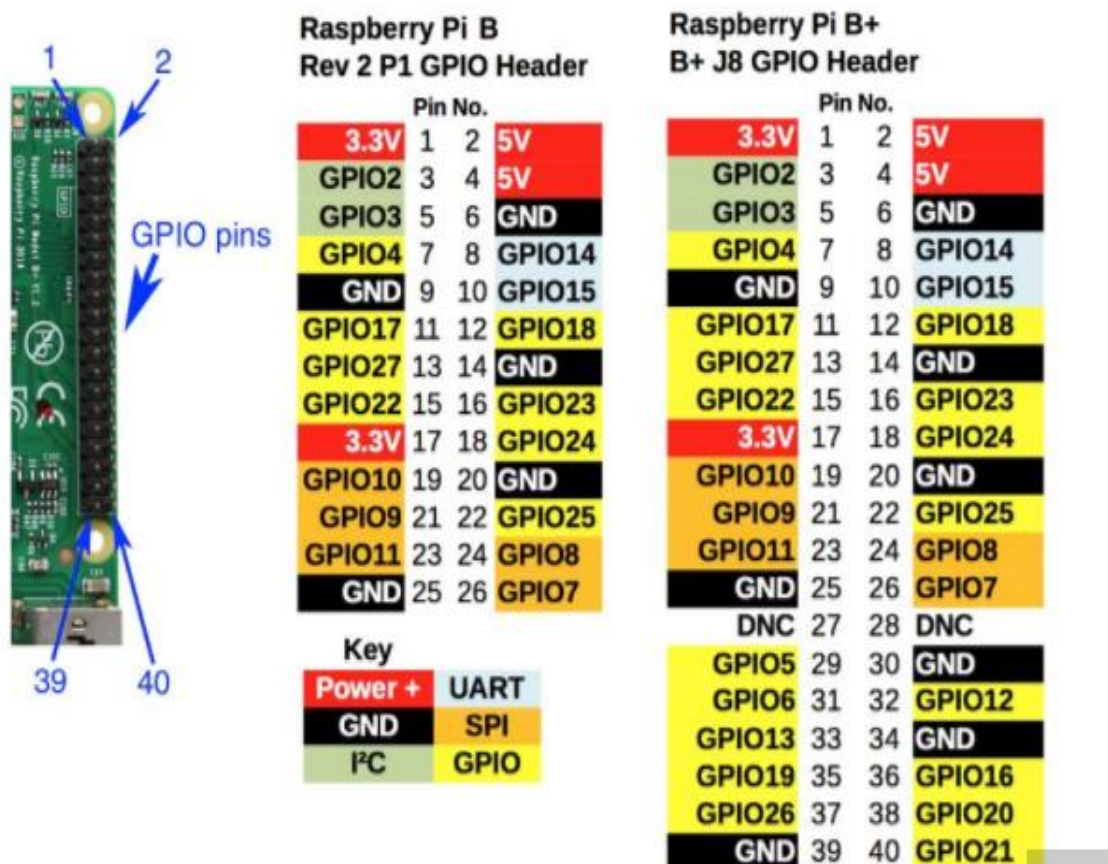


Figure 3.3.2: Pin Diagram of Raspberry Pi

B. Ultrasonic sensor:

The Ultrasonic Sensors belongs to a category of sensors that emits ultrasound i.e. sound of frequency more than 20 kHz. Initially, a *trigger pulse* is given as an input to the ultrasonic sensor using Raspberry Pi.

The ultrasonic sensor then emits a short 40 kHz ultrasonic burst signal. This burst signal travels through the air at approximately 343ms⁻¹, hits an object and then bounces back to the sensor resulting in an *output pulse*.

This output pulse is captured by Raspberry Pi. Then, using the time taken by the pulse to return back, we calculate the distance from the obstacle.

The sensor consists of four pins: (1) VCC, (2) Trigger,(3) Echo and (4) Ground as illustrated in Figure 3.3.3,

- **VCC** - It is used to provide 5V power to the sensor.
- **Trigger (Trig)** - Takes in Input Pulse to trigger the sensor.
- **Echo** - It is used to receive the Output Pulse i.e. the echo from the object detected.
- **Ground (GND)** - It connects sensor to the ground.



Figure 3.3.3: Ultrasonic Sensor

C. Android Application:

The Android application consists of two modes i.e. Automatic and manual, the user can switch into. Depending on the requirement, the user can use the manual mode or the automatic mode. The manual mode responds with the distance between the obstacle and the user only when they click the button. Whereas, in the automatic mode, the user will be receiving the distance at regular intervals of time.

The user has the option to switch between both the modes. Whenever a button is pressed, the TTS module efficiently responds with a message so as to let the user know which mode they have selected.

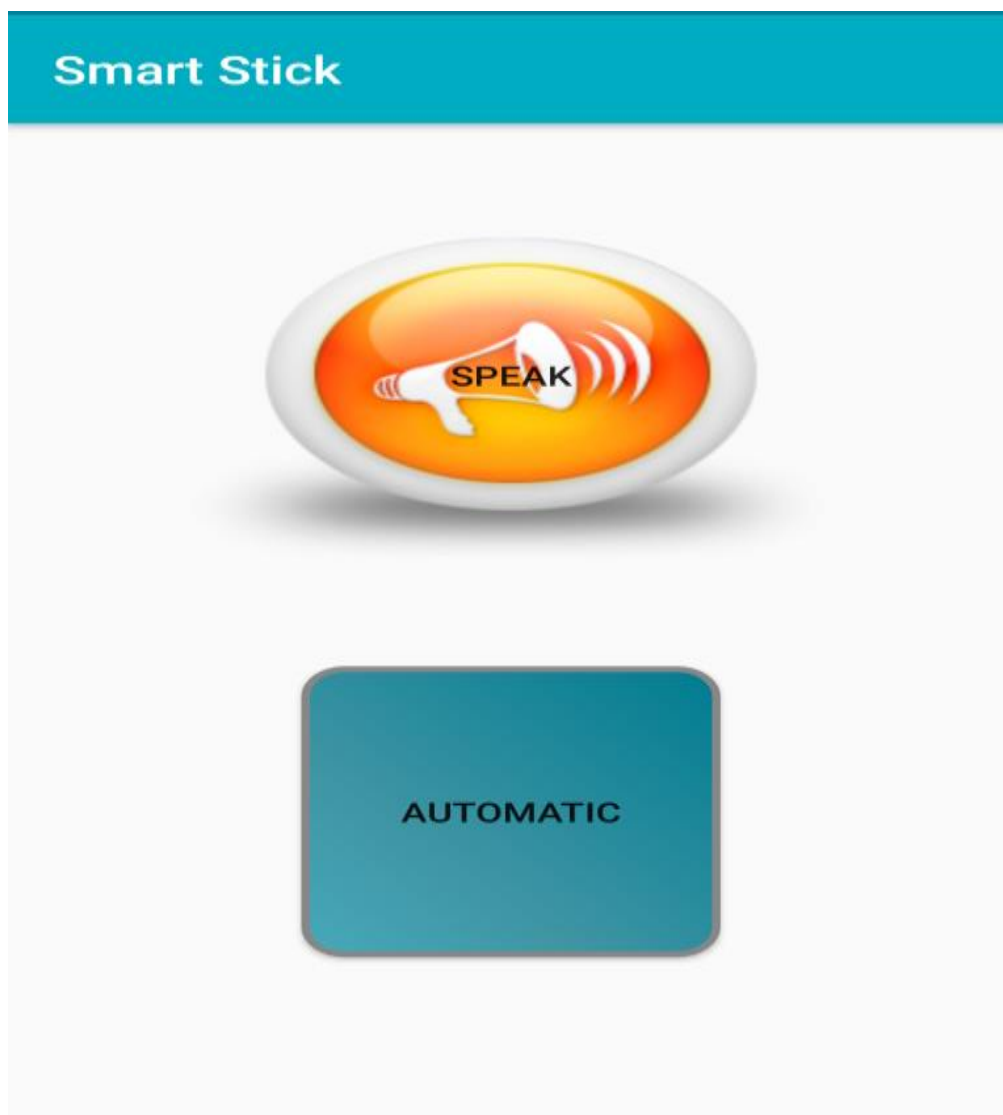


Figure 3.3.4: Smart Stick Android Application

D. Firebase Database:

The Google Firebase real-time database is a cloud hosted database that supports multiple platforms Android, iOS and Web. All the data is stored in JSON format and any changes in data, reflects immediately by performing sync across all the platforms & devices. This allows us to build more flexible real-time apps easily with minimal effort. The Firebase Real-time Database also lets you build rich, collaborative applications by allowing secure access to the database directly from client-side code. Data is persisted locally, and even while offline, real-time events continue to fire, giving the end user a responsive experience. When the device regains connection, the real-time Database synchronizes the local data changes with the remote updates that occurred while the client was offline, merging any conflicts automatically. When the distance is calculated from the ultrasonic sensor's pulse signals, the shortest distance among the sensors is sent into the firebase database. This distance is then retrieved by the android application. We can retrieve the value of the distance stored in the database.

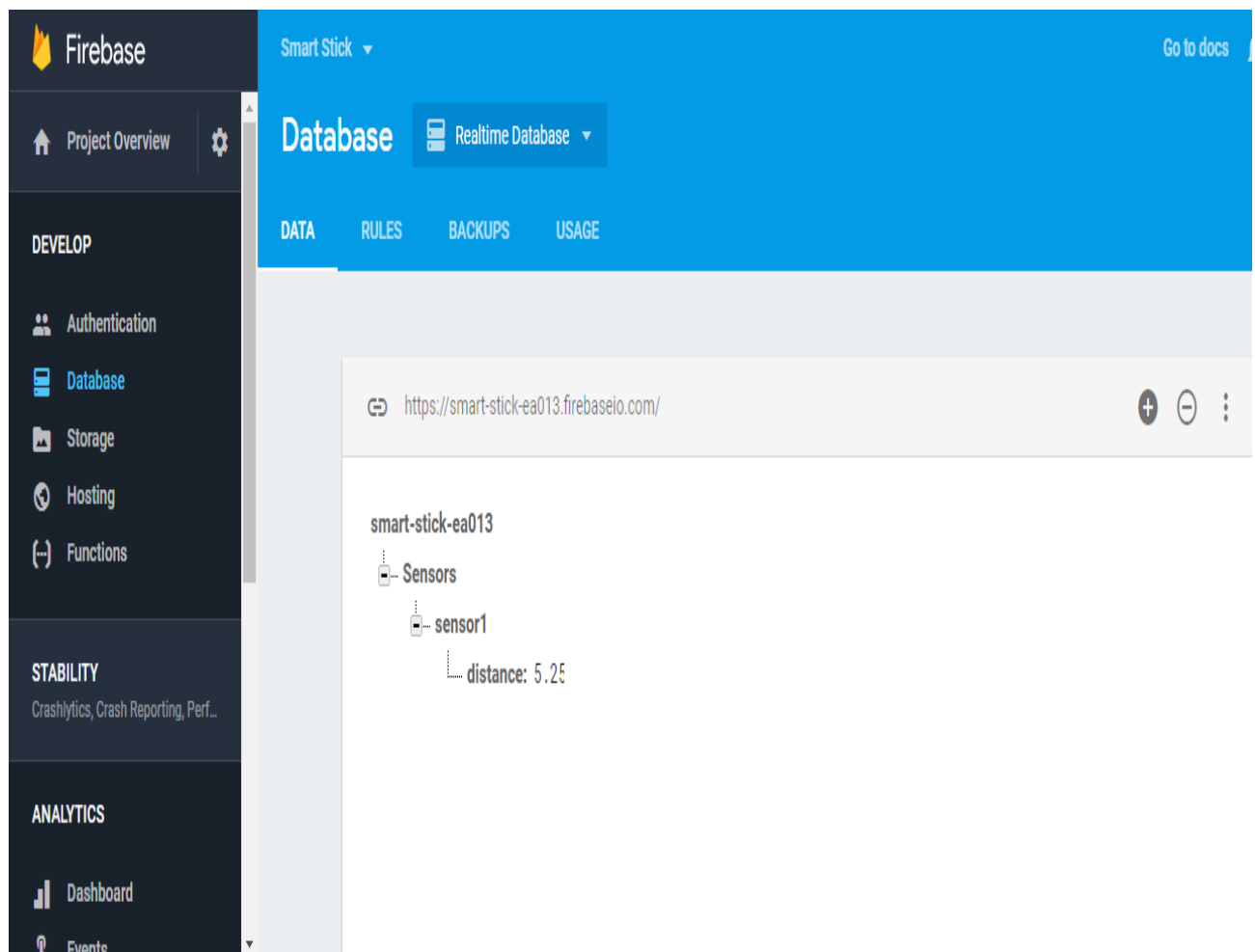


Figure 3.3.5: Firebase Database

The working of the system involves the following functionalities:

1. Distance Calculation
2. Posting and retrieving value from firebase database.
3. Text to Speech Conversion.
4. Modes of operation

Distance Calculation:

The ultrasonic sensors emit an ultrasound at 40000 Hz which travels through the air and if there is an object on its path, it will bounce back to module. The transmitter transmits short bursts which get reflected by target and are picked up the receiver.

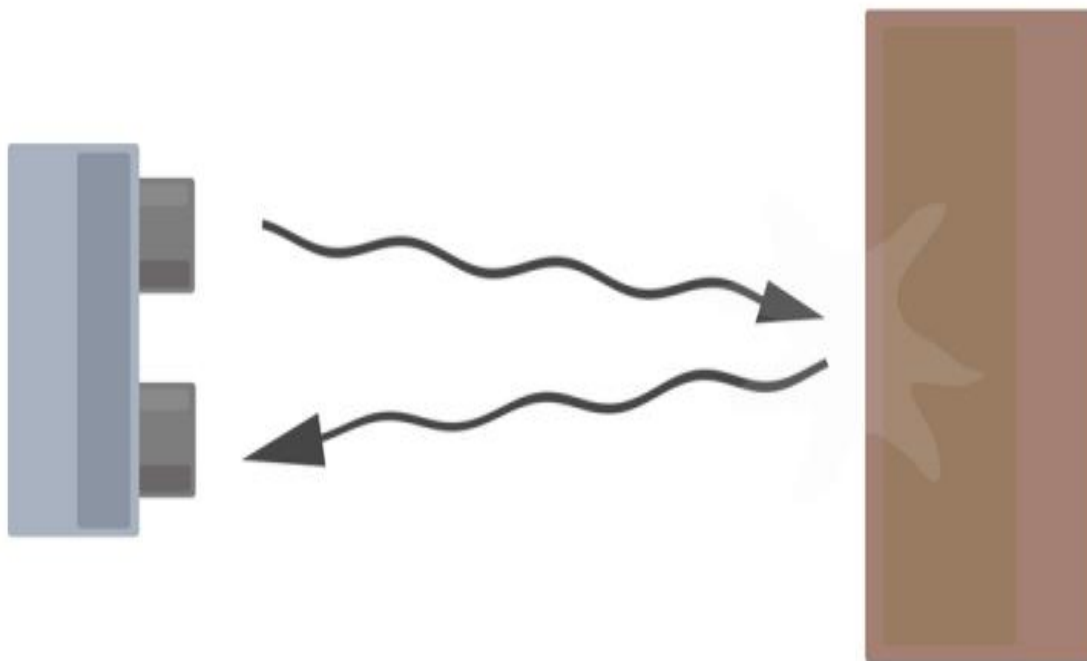


Figure 3.3.6: TRIG and ECHO of Ultrasonic Sensors

The time difference between transmission and reception of ultrasonic signals is calculated. Considering the travel time and the speed of the sound you can calculate the distance.

Here to find the distance between the obstacle and the person, we use Distance Formula:

$$\text{distance} = \text{speed} * \text{time}$$

$$\text{OD} = \{[\text{Speed of Sound} * \text{Time Taken}] / 2\}$$

Where,

- **OD:** Distance between an obstacle and the person in meters.
- **Speed of Sound:** We take speed of sound as 343meter/sec.
- **Time Taken:** It is the time interval between the pulse emitted and the pulse received.
- Since, the time taken by the pulse is twice the distance travelled; we divide the equation by 2.

```
while GPIO.input(ECHO2)==0:           #Check whether the ECHO is LOW
    pulse_start2 = time.time()         #Saves the last known time of LOW pulse

while GPIO.input(ECHO2)==1:           #Check whether the ECHO is HIGH
    pulse_end2 = time.time()           #Saves the last known time of HIGH pulse

pulse_duration2 = pulse_end2 - pulse_start2 #Get pulse duration to a variable

distance2 = pulse_duration2 * 17150    #Multiply pulse duration by 17150 to get distance
distance2 = round(distance2, 2)        #Round to two decimal points

#Comparing 2 sensor's distance
```

Figure 3.3.7: Distance Calculation Program for Smart Stick

Posting and retrieving value from firebase database:

After the distance is successfully calculated, we send the value of the distance to the firebase database. Here we make use of python programming language to accomplish to post values to the database. First, the distance obtained from each of the two ultrasonic sensors is compared with each other.

Then, the shortest distance between the two values is posted to database. If both the values exceed 400cm, then the value posted to the database is null. As long as the sensors are running, the calculated, compared and sent to database continuously.

In the database, the old value of the distance is replaced with the new reading every time the values are posted. This process is dynamic and the values are updated simultaneously with minimal delay.

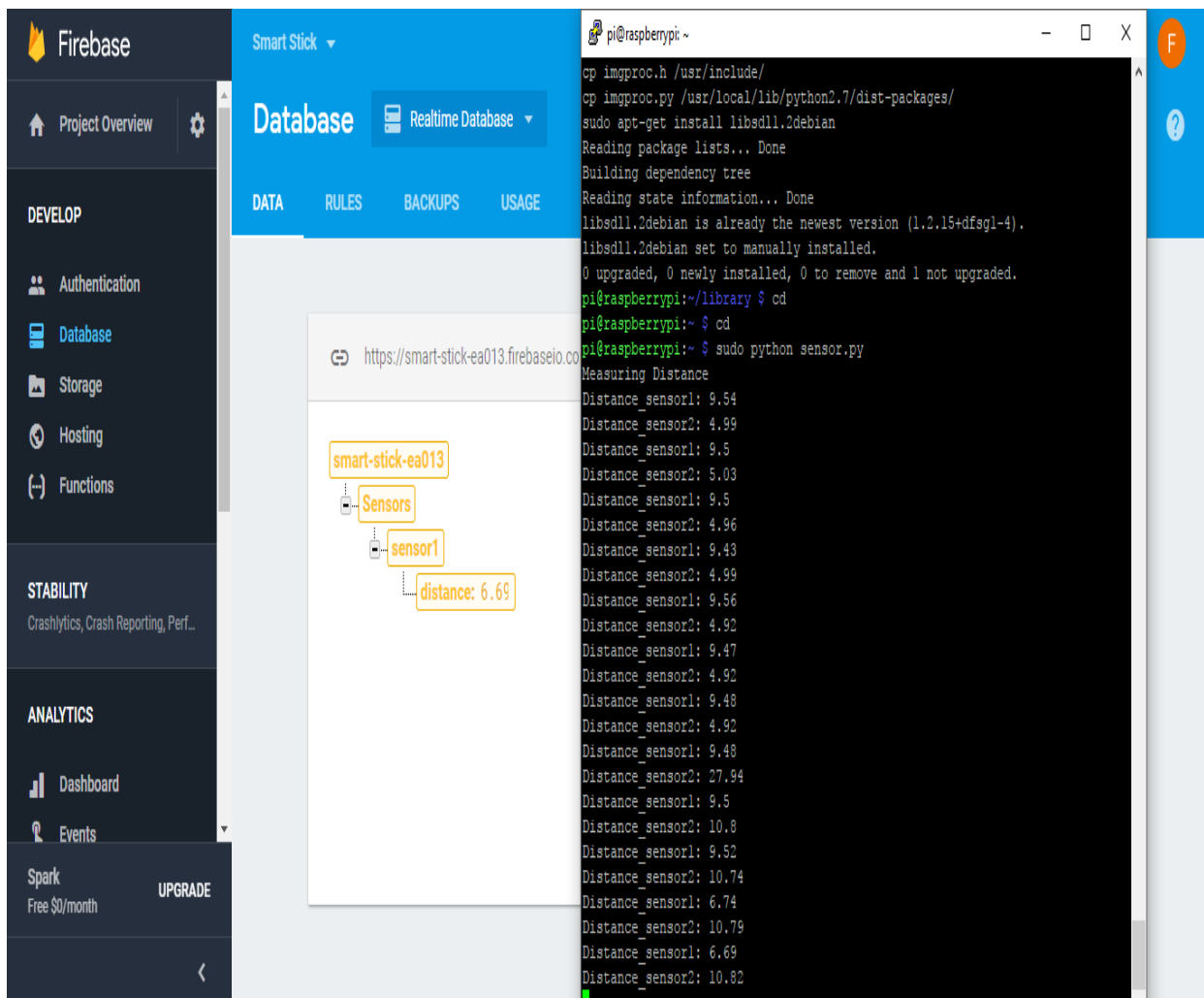


Figure 3.3.8: Posting distance to Firebase Database

After we have successfully posted the reading from the sensors to the database, we now have to retrieve the data and run it in our android application.

In order to retrieve the value of the distance from the database to the application, we use java programming language. We use data snapshots of the database to efficiently retrieve the data.

```
@Override
public void run() {
    //Called each time when 1000 milliseconds (1 second) (the period parameter)
    ref.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            distance= dataSnapshot.child("sensor1").getValue().toString();
        }

        @Override
        public void onCancelled(DatabaseError databaseError) {

        }
    });
}
```

Figure 3.3.9: Retrieving from Firebase Database

After retrieving the distance, this value is stored as a string variable in the application. The text-to-speech module synthesizes this string to an audio file.

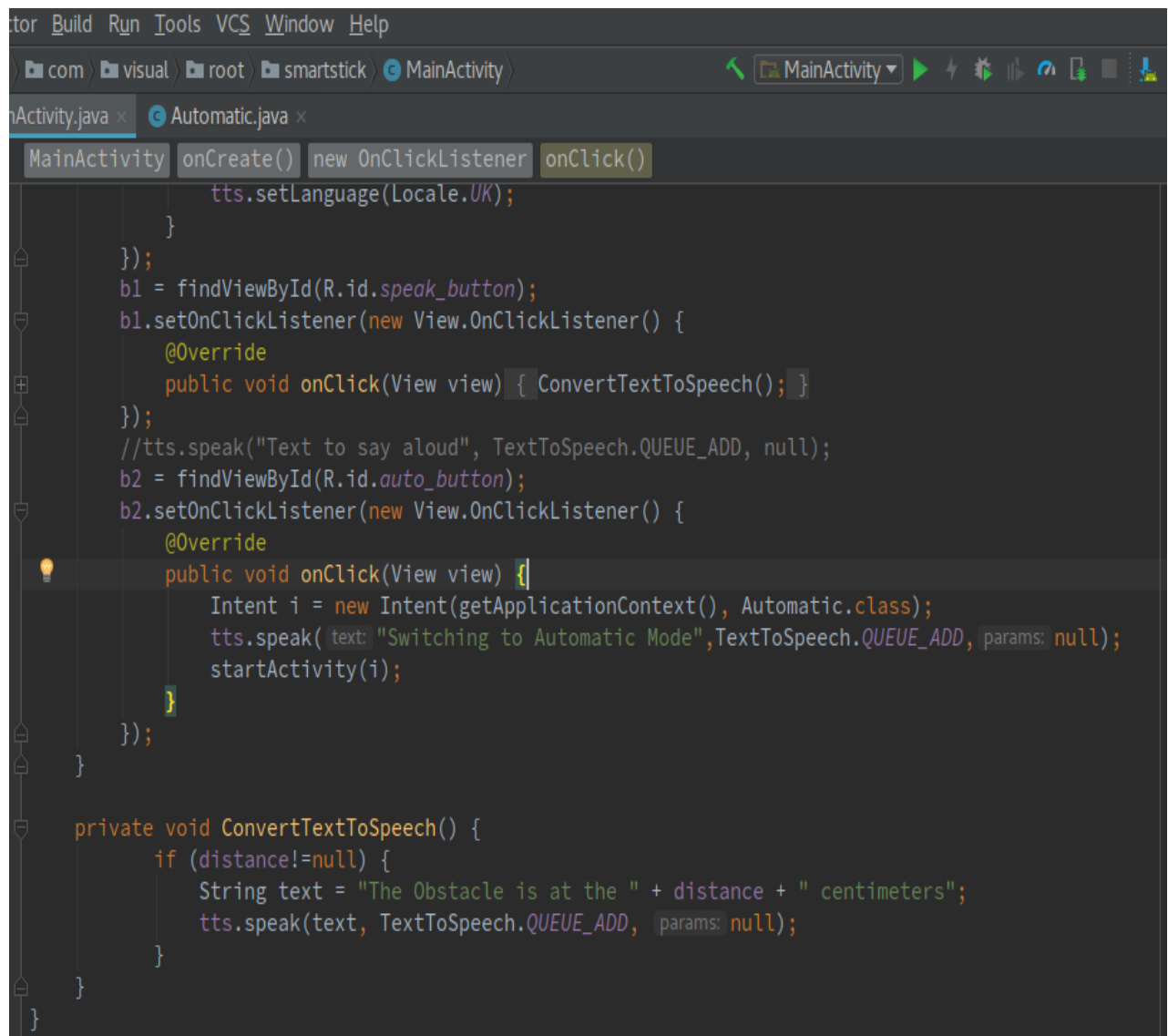
Text-to-Speech Conversion:

Android allows you convert your text into voice. Not only you can convert it but it also allows you to speak text in variety of different languages. In order to accomplish this, we are using the Text-to-Speech (TTS) module.

The TTS module synthesizes speech from text for immediate playback or to create a sound file. This sound file is run on the application for the user to listen.

After retrieving the value of distance from the database, it is put into a string which is synthesized for immediate playback. Depending on the mode of operation, the TTS module is run.

In the automatic mode, the TTS module runs every five seconds. In the Manual mode, the TTS module is activated only when the user presses the speak button.



```
MainActivity.java Automatic.java x
MainActivity onCreate() new OnClickListener onClick()
    tts.setLanguage(Locale.UK);
}
});
b1 = findViewById(R.id.speak_button);
b1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) { ConvertTextToSpeech(); }
});
//tts.speak("Text to say aloud", TextToSpeech.QUEUE_ADD, null);
b2 = findViewById(R.id.auto_button);
b2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent i = new Intent(getApplicationContext(), Automatic.class);
        tts.speak( text: "Switching to Automatic Mode",TextToSpeech.QUEUE_ADD, params: null);
        startActivity(i);
    }
});
}

private void ConvertTextToSpeech() {
    if (distance!=null) {
        String text = "The Obstacle is at the " + distance + " centimeters";
        tts.speak(text, TextToSpeech.QUEUE_ADD, params: null);
    }
}
}
```

Figure 3.3.10: TTS Code for Smart Stick Application

Modes of operation:

The Smart Stick device consists of two modes namely manual and automatic. These modes can be switched according to the user's requirement with the help of the application.

The manual mode is the default mode of the application. In this, the speech is activated only when user presses the “speak” button. The speech is activated only once. If the user presses the button again, it again responds back in the form of speech. For example, when the speak button is pressed, it gives a response “The obstacle is at 120cm”.

If the user wishes to switch the mode to automatic mode, then he/she simply has to press the bottom button which takes us to the automatic mode activity. In order to let the user navigate the application with ease, whenever the user clicks on the automatic button they hear a speech “Switching to automatic mode”.

In the automatic mode, we have two buttons namely start and stop. When the user presses the start button the automatic mode is activated and response is given to the user at every five seconds about the obstacles in front of them in the form of speech.

The data is continuously retrieved from the database in this mode and converted to speech simultaneously. The user can stop the automatic mode by pressing the stop button which gives the response “Press again to switch to manual mode”.

The user can either choose to activate the automatic mode again by clicking the start button or switch back to manual mode by clicking the stop button.

3.4 Modules

We have divided our project into four modules respectively:

1. Raspberry Pi 1 Model B+.
2. Ultrasonic Sensors.
3. Google Firebase.
4. Android Application.

Module 1 RASPBERRY PI

Raspberry Pi is a credit card sized single board, low cost computer. It takes input from the GPIO pins, which can be attached to LEDs, switches, analog signals and other devices. For our proposed design, we connect the GPIO pins to the ultrasonic sensors. It requires a power

source of 5V to be operational and we have to insert a Micro SD memory card in it, which acts as its permanent memory.

For our design Raspberry Pi 1 Model B+ is used. It contains 4 USB ports, a HDMI port, an audio jack port and an Ethernet port. The Ethernet port helps the device connect to the Internet and install required driver APIs. It has a 700 MHz single core processor and supports programming languages such as Python, Java, C, and C++ etc.

This minicomputer runs our algorithm, which helps to calculate the distance from the obstacle based on the input it receives from the sensors. We can use a battery, portable charger, micro USB or a rectifier as the input power source. The Raspberry Pi primarily uses Raspbian, a Debian-based Linux operating system.



Figure.3.4.1: Raspberry Pi 1 Model B+

Module 2 Ultrasonic Sensor

The Ultrasonic Sensors belongs to a category of sensors that emits ultrasound i.e. sound of frequency more than 20 kHz. Initially, a *trigger pulse* is given as an input to the ultrasonic sensor using Raspberry Pi.

The ultrasonic sensor then emits a short 40 kHz ultrasonic burst signal. This burst signal travels through the air at approximately 343ms⁻¹, hits an object and then bounces back to the sensor resulting in an *output pulse*.

This output pulse is captured by Raspberry Pi. Then, using the time taken by the pulse to return back, we calculate the distance from the obstacle.



Figure.3.4.2: Ultrasonic Sensor

The sensor consists of four pins: (1) VCC, (2) Trigger, (3) Echo and (4) Ground as illustrated in Figure 3.4.2.

- **VCC** - It is used to provide 5V power to the sensor.
- **Trigger (Trig)** - Takes in Input Pulse to trigger the sensor.
- **Echo** - It is used to receive the Output Pulse i.e. the echo from the object detected.
- **Ground (GND)** - It connects sensor to the ground.

MODULE 3 Smart Stick Application

The Android application consists of two modes i.e. Automatic and manual, the user can switch into. Depending on the requirement, the user can use the manual mode or the automatic mode. The manual mode responds with the distance between the obstacle and the user only when they click the button.

Whereas, in the automatic mode, the user will be receiving the distance at regular intervals of time.

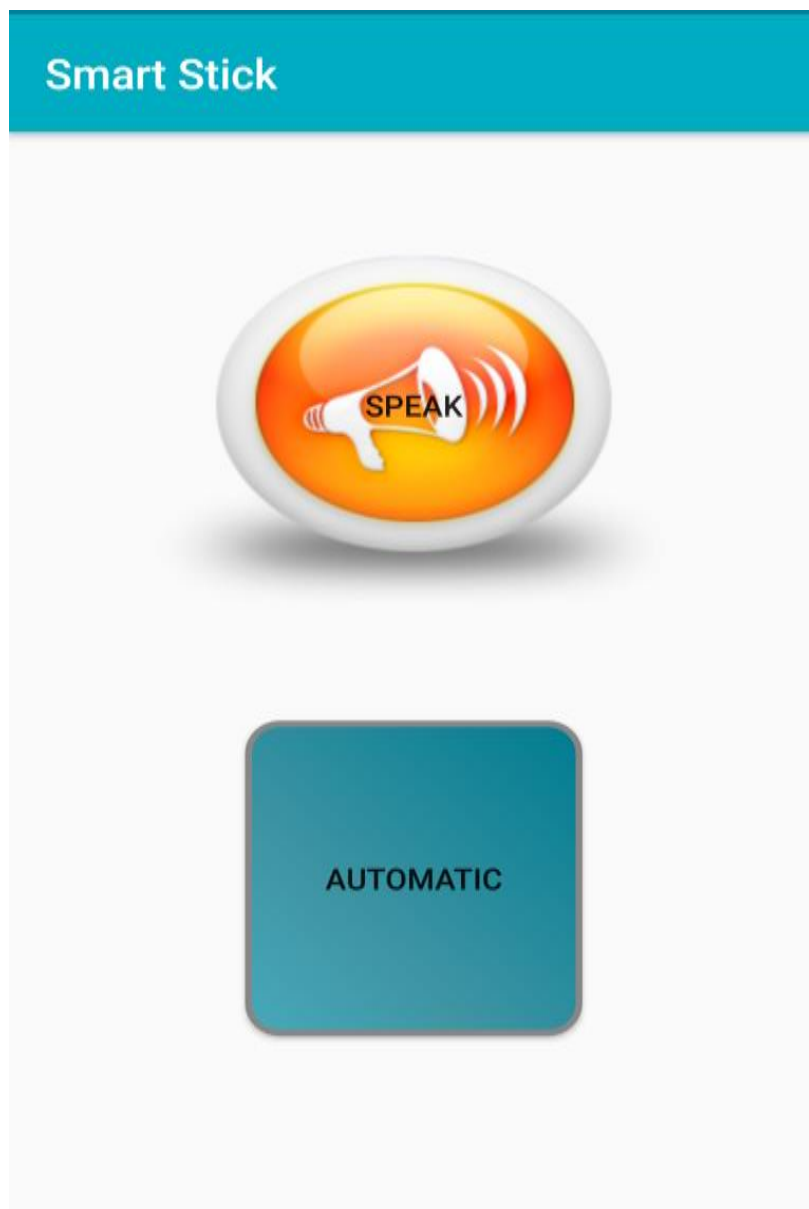


Figure.3.4.3: Android Application of Smart Stick

MODULE 4 Firebase Database

The Google Firebase real-time database is a cloud hosted database that supports multiple platforms Android, iOS and Web. All the data is stored in JSON format and any changes in data, reflects immediately by performing sync across all the platforms & devices. This allows us to build more flexible real-time apps easily with minimal effort.

The Firebase Real-time Database also lets you build rich, collaborative applications by allowing secure access to the database directly from client-side code.

Data is persisted locally, and even while offline, real-time events continue to fire, giving the end user a responsive experience. When the device regains connection, the real-time Database synchronizes the local data changes with the remote updates that occurred while the client was offline, merging any conflicts automatically. When the distance is calculated from the ultrasonic sensor's pulse signals, the shortest distance among the sensors is sent into the firebase-database.

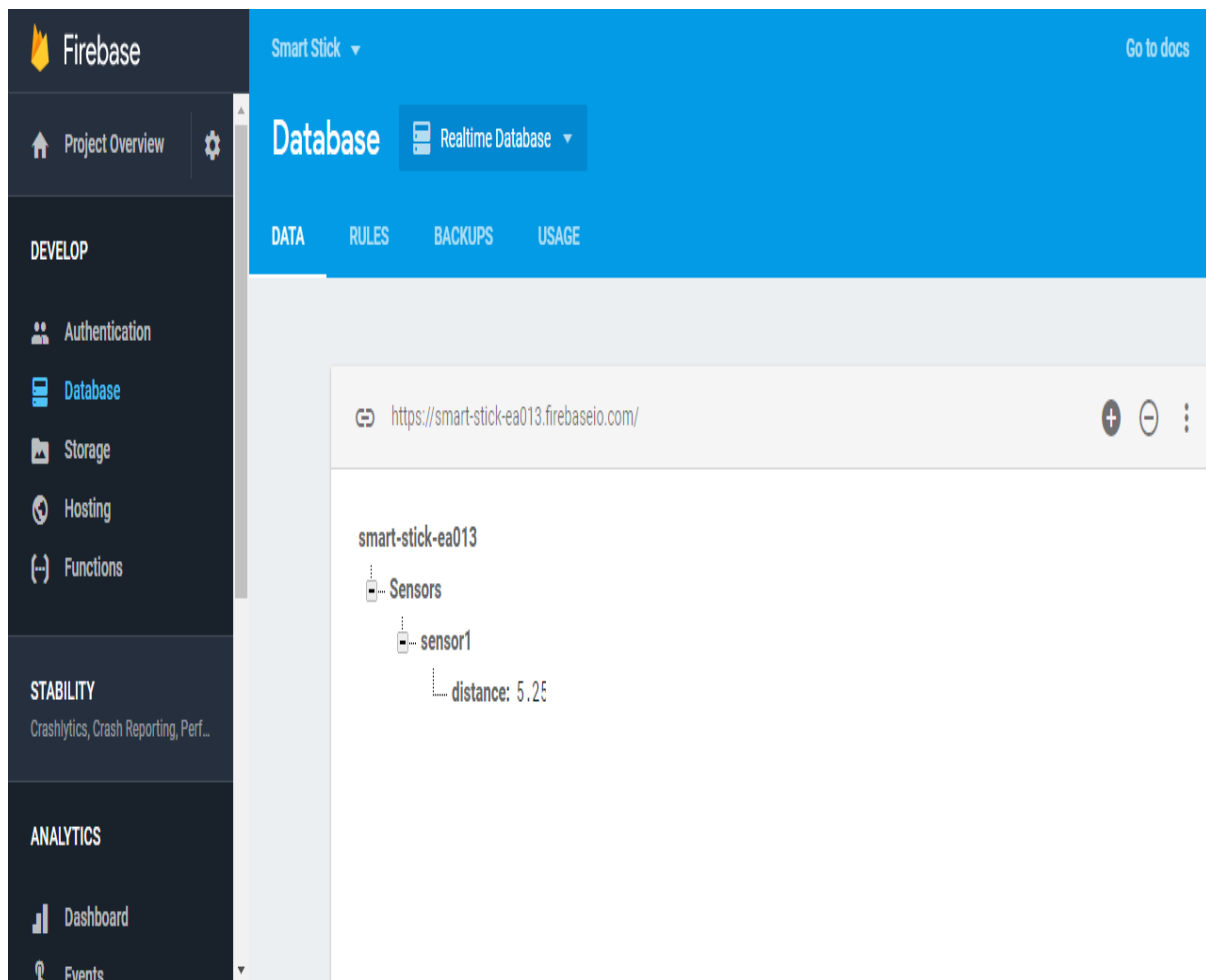


Figure 3.4.4: Firebase Database

This distance is then retrieved by the android application. In Figure 4.1.4, we can retrieve the value of the distance stored in the database.

3.5 UML Diagrams

The Unified Modeling Language (UML) is a standard language for writing software blueprints. The UML is a language for

- Visualizing
- Specifying
- Constructing
- Documenting the artifacts of a software intensive system.

The UML is a language which provides vocabulary and the rules for combining the words in that vocabulary for the purpose of communication. A modeling language is a language whose vocabulary and the rules focus on the conceptual and physical representation of a system. Modelling yields an understanding of a system.

UML is a way of visualizing a software program using a collection of diagrams. The notation has evolved from the work of Grady Booch, James Rumbaugh, Ivar Jacobson, and the Rational Software Corporation to be used for object-oriented design, but it has since been extended to cover a wider variety of software engineering projects. Today, UML is accepted by the Object Management Group (OMG) as the standard for modeling software development.

The Unified Modeling Language (UML) was created to forge a common, semantically and syntactically rich visual modeling language for the architecture, design, and implementation of complex software systems both structurally and behaviourally. UML has applications beyond software development, such as process flow in manufacturing. It is analogous to the blueprints used in other fields, and consists of different types of diagrams.

In the aggregate, UML diagrams describe the boundary, structure, and the behaviour of the system and the objects within it. UML is not a programming language but there are tools that can be used to generate code in various languages using UML diagrams. UML has a direct relation with object-oriented analysis and design.

There are many problem-solving paradigms or models in Computer Science, which is the study of algorithms and data. There are four problem-solving model categories: imperative, functional, declarative and object-oriented languages (OOP). In object-oriented languages, algorithms are expressed by defining ‘objects’ and having the objects interact with each other. Those objects are things to be manipulated and they exist in the real world.

They can be buildings, widgets on a desktop, or human beings. Object-oriented languages dominate the programming world because they model real-world objects. UML is a combination of several object-oriented notations: Object-Oriented Design, Object Modeling Technique, and Object-Oriented Software Engineering.

UML uses the strengths of these three approaches to present a more consistent methodology that's easier to use. UML represents best practices for building and documenting different aspects of software and business system modeling. The objects in UML are real world entities that exist around us.

In software development, objects can be used to describe, or model, the system being created in terms that are relevant to the domain. Objects also allow the decomposition of complex systems into understandable components that allow one piece to be built at a time.

3.5.1 Class Diagram

Classes are typically modelled as rectangles with three sections: the top section for the name of the class, the middle section for the attributes of the class, the bottom section for the methods of the class.

The most commonly used UML diagram, and the principal foundation of any object-oriented solution. Classes within a system, attributes and operations and the relationship between each class. Classes are grouped together to create class diagrams when diagramming large systems.

Attributes are the information stored about an object, while methods are the things an object or class do.

The given diagram depicts the class diagram of the application.

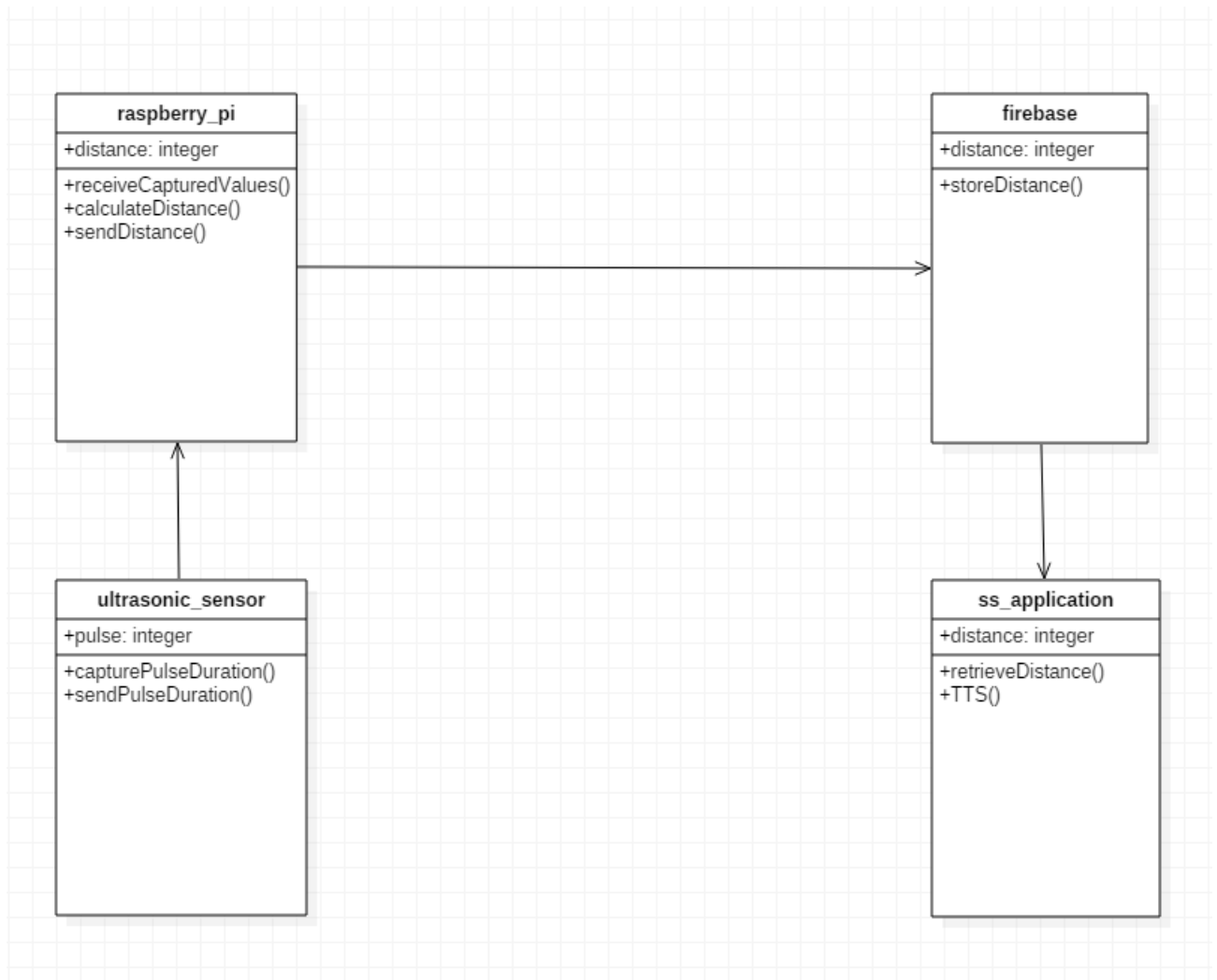


Figure 3.5.1.1: Class Diagram for Smart Stick

3.5.2 Activity Diagram

Activity diagrams represent the business and operational and the workflows of a system. Graphically represented business or operational workflows to show the activity of any part or component in the system. Activity diagrams are used as an alternative to State Machine diagrams.

An Activity diagram is a dynamic diagram that shows the activity and the event that causes the object to be in the particular state.

The given diagram depicts the activity diagram of the application.

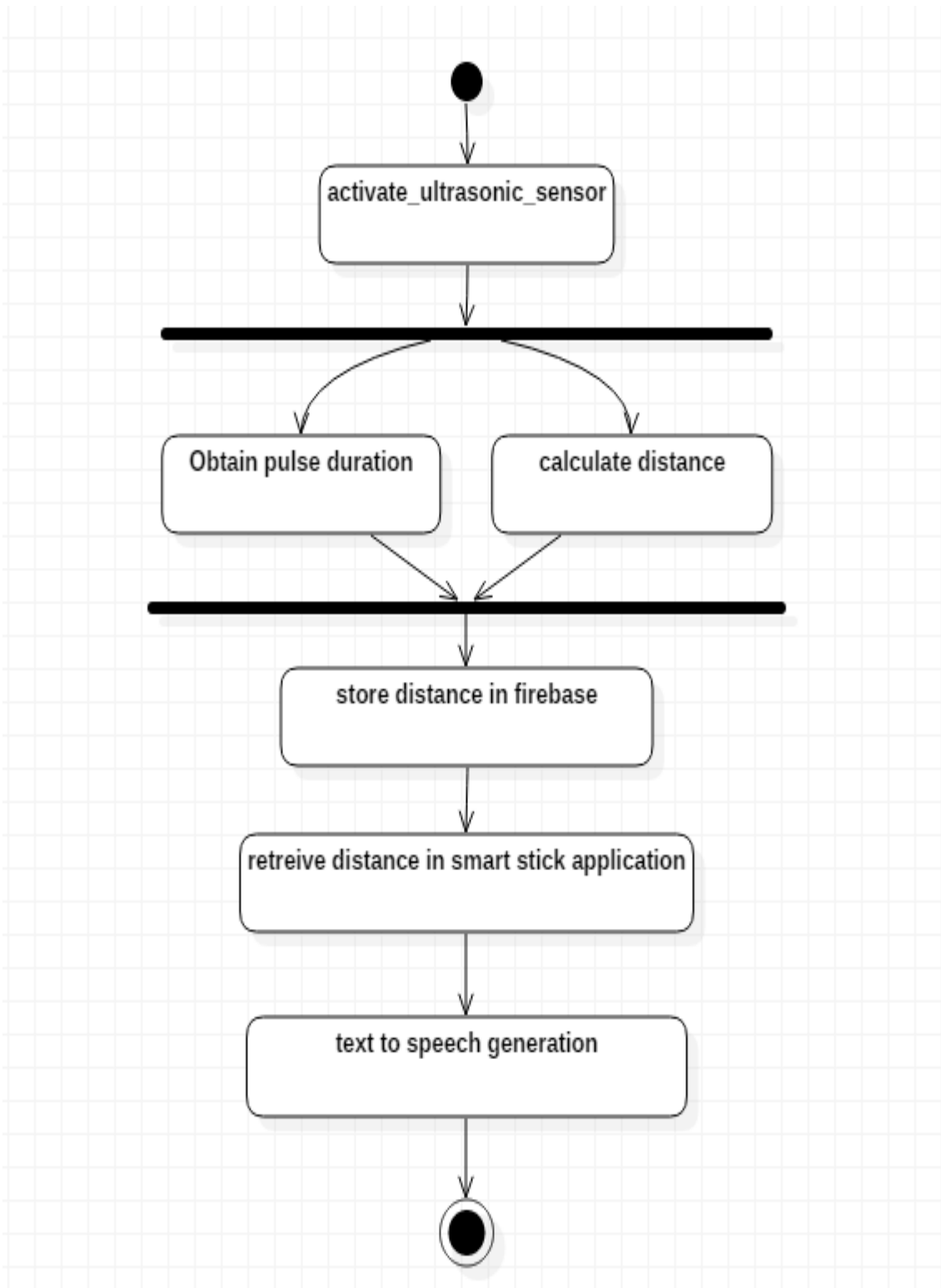


Figure 3.5.2.1: Activity Diagram for Smart Stick

3.5.3 Sequence Diagram

UML sequence diagrams are used to represent the flow of messages, events and actions between the objects or components of a system. Time is represented in the vertical direction showing the sequence of interactions of the header elements, which are displayed horizontally at the top of the diagram.

The given diagram depicts the sequence diagram of the application.

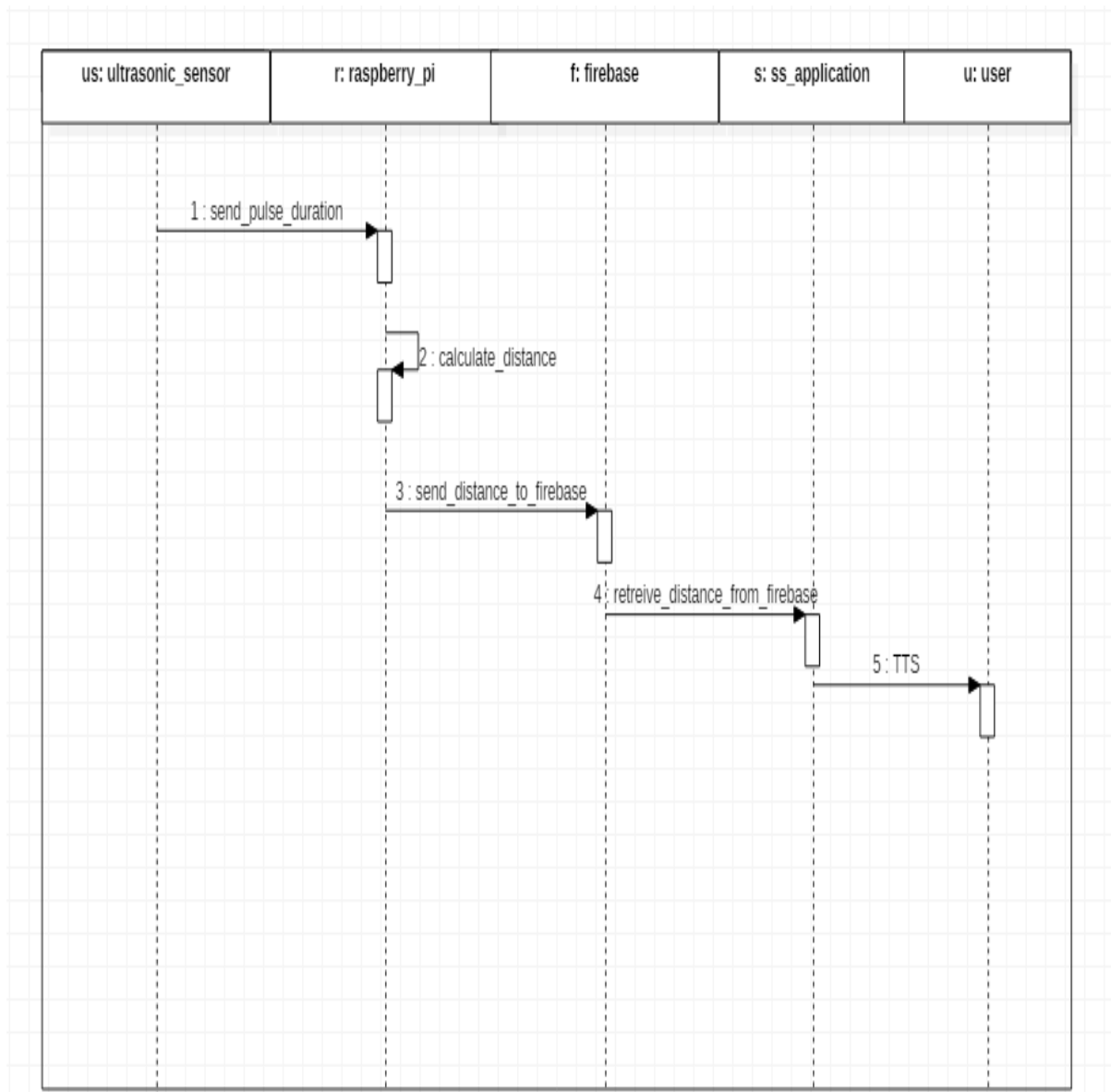


Figure 3.5.3.1: Sequence Diagram for Smart Stick

3.5.4 Collaboration Diagram

The second interaction diagram is the collaboration diagram. It shows the object organization as seen in the following diagram. In the collaboration diagram, the method call sequence is indicated by some numbering technique. The number indicates how the methods are called one after another.

Method calls are similar to that of a sequence diagram. However, difference being the sequence diagram does not describe the object organization, whereas the collaboration diagram shows the object organization.

The given diagram depicts the collaboration diagram of the application.

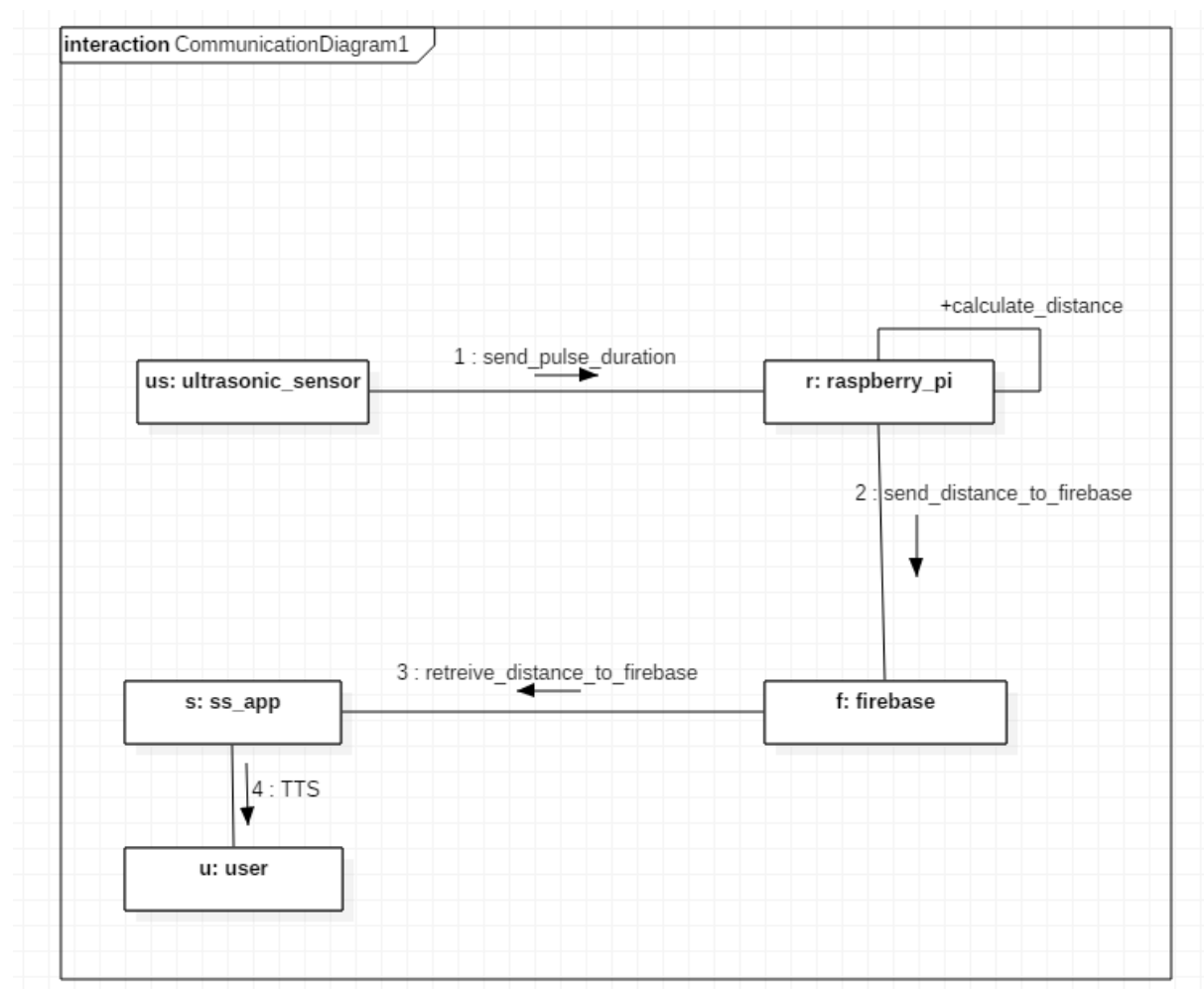


Figure 3.5.4.1: Collaboration diagram for Smart Stick

3.5.5 State Chart Diagram

A State diagram shows the different states an object is in during the lifecycle of its existence in the system, and the transitions in the states of the objects. Graphically represented business or operational workflows to show the activity of any part or component in the system. Activity diagrams are used as an alternative to State chart diagrams.

These transitions depict the activities causing these transitions, shown by arrows.

The given diagram depicts the state-chart diagram of the application.

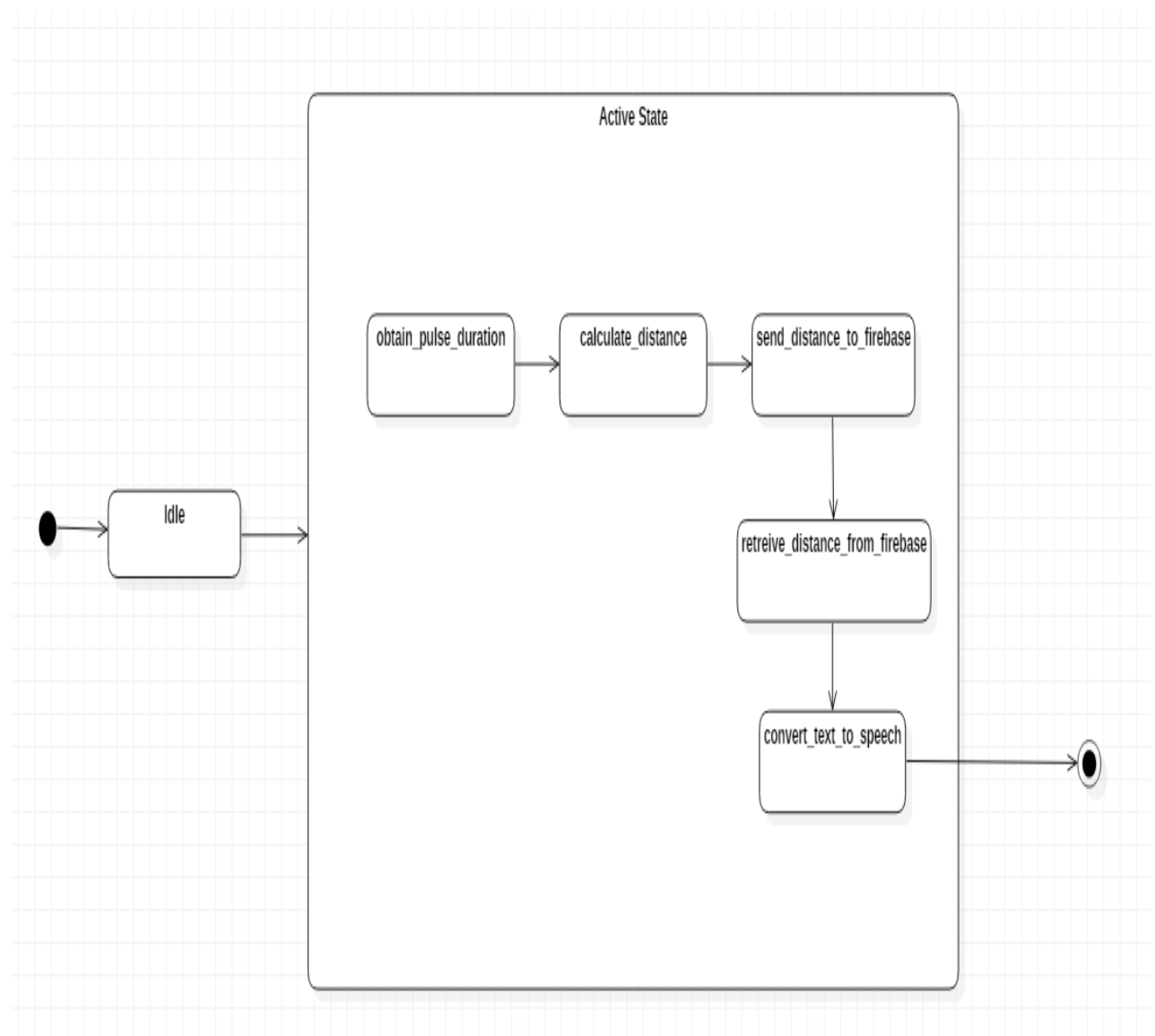


Figure 3.5.5.1: State Chart Diagram for Smart Stick

4. IMPLEMENTATION

4.1 Sample code

Sensor.py

```
import RPi.GPIO as GPIO          #Import GPIO library

import time

import atexit

import datetime

from firebase import firebase

import urllib2, urllib, httplib

import json

import os

from functools import partial      #Import time libra$

GPIO.setmode(GPIO.BCM)           #Set GPIO pin numbering

firebase = firebase.FirebaseApplication('https://smart-stick-67s56.firebaseio.com')

TRIG = 20                         #Associate pin 38 to TRIG

ECHO = 26                         #Associate pin 37 to ECHO

TRIG2 = 16                        #Associate pin 36
to TRIG2

ECHO2 = 19                        #Associate pin 35 to ECHO2

firebase.put('Users/S0001',"test","Measuring Distance")

print "Measuring Distance"

GPIO.setup(TRIG,GPIO.OUT)         #Sensor1 #Set pin as GPIO out

GPIO.setup(ECHO,GPIO.IN)          #Set pin as GPIO in

GPIO.setup(TRIG2.GPIO.OUT)        #Sensor2 #Set pin as GPIO out
```

```
GPIO.setup(ECHO2,GPIO.IN)
```

```
#Set pin as GPIO in
```

```
while True:
```

```
    #Sensor 1
```

```
    GPIO.output(TRIG, False)          #Set TRIG as LOW
```

```
    print "Waiting For Sensor To Settle"
```

```
    time.sleep(2)                      #Delay of 2 seconds
```

```
    GPIO.output(TRIG, True)           #Set TRIG as HIGH
```

```
    time.sleep(0.00001)               #Delay of 0.00001 seconds
```

```
    GPIO.output(TRIG, False)          #Set TRIG as LOW
```

```
    while GPIO.input(ECHO)==0:         #Check whether the ECHO is LOW
```

```
        pulse_start = time.time()      #Saves the last known time of LOW pulse
```

```
    while GPIO.input(ECHO)==1:         #Check whether the ECHO is HIGH
```

```
        pulse_end = time.time()        #Saves the last known time of HIGH pulse
```

```
    pulse_duration = pulse_end - pulse_start #Get pulse duration to a variable
```

```
    distance1 = pulse_duration * 17150   #Multiply pulse duration by 17150 to get distance
```

```
    distance1 = round(distance1, 2)      #Round to two decimal points
```

```
    #Sensor 2
```

```
    GPIO.output(TRIG2, False)          #Set TRIG as LOW
```

```
    print "Waiting For Sensor To Settle"
```

```
    time.sleep(2)                      #Delay of 2 seconds
```

```
    GPIO.output(TRIG2, True)           #Set TRIG as HIGH
```

```
    time.sleep(0.00001)               #Delay of 0.00001 seconds
```

```

GPIO.output(TRIG2, False)          #Set TRIG as LOW

while GPIO.input(ECHO2)==0:        #Check whether the ECHO is LOW

    pulse_start2 = time.time()     #Saves the last known time of LOW pulse

while GPIO.input(ECHO2)==1:        #Check whether the ECHO is HIGH

    pulse_end2 = time.time()       #Saves the last known time of HIGH pulse

pulse_duration2 = pulse_end2 - pulse_start2 #Get pulse duration to a variable

distance2 = pulse_duration2 * 17150 #Multiply pulse duration by 17150 to get distance

distance2 = round(distance2, 2)    #Round to two decimal points

#Comparing 2 sensor's distance

if distance1 < distance2 and distance1 < 400: #Check whether the distance is within range

print "Distance:",distance1 - 0.5,"cm" #Print distance with 0.5 cm calibration

    firebase.put('Users/S0001',"distance",distance1)

elif distance2 < distance1 and distance2 < 400:

print "Distance:",distance2 - 0.5,"cm" #Print distance with 0.5 cm calibration

firebase.put('Users/S0001',"distance",distance2) #firebase.post('/Sensors/sensor1', distance)

else:

print "Out Of Range"              #display out of range

```


MainActivity.java

```
package com.visual.root.smartstick;

import android.content.Intent;

import android.content.SharedPreferences;

import android.speech.tts.TextToSpeech;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.support.v7.widget.Toolbar;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.Button;

import android.widget.Toast;


import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

import java.util.Locale;
```

```

public class MainActivity extends AppCompatActivity {

    TextToSpeech tts;

    DatabaseReference ref;

    SharedPreferences pref;

    Button b1, b2;

    String distance, ssid;

    View.OnClickListener listener;

    FirebaseAuth firebaseAuth;

    @Override

    public boolean onCreateOptionsMenu(Menu menu) {

        getMenuInflater().inflate(R.menu.toolbar_menu, menu);

        return true;

    }

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        Intent i = getIntent();

        ssid = i.getStringExtra("smart_id");

        Toolbar myToolbar = findViewById(R.id.my_toolbar);

        setSupportActionBar(myToolbar);

```

```

        //firebaseAuth = FirebaseAuth.getInstance();

ref      =      FirebaseDatabase.getInstance().getReferenceFromUrl("https://smart-stick-
ea013.firebaseio.com/Users");

rpref = getSharedPreferences("DATA", 0);

ssid = rpref.getString("ssid", "");

tts = new TextToSpeech(MainActivity.this, new TextToSpeech.OnInitListener() {

        @Override

public void onInit(inti) {

if (i == TextToSpeech.SUCCESS) {

tts.setLanguage(Locale.UK);

        }

        }

});

b1 = findViewById(R.id.speak_button);

b2 = findViewById(R.id.auto_button);

b1.setOnClickListener(new View.OnClickListener() {

        @Override

public void onClick(View view) {

ConvertTexttoSpeech();

        }

});

b2.setOnClickListener(new View.OnClickListener() {

        @Override

public void onClick(View view) {

```

```

        Intent i = new Intent(getApplicationContext(), Automatic.class);

        i.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);

        tts.speak("Switching to Automatic Mode", TextToSpeech.QUEUE_ADD, null);

        i.putExtra("smart_id", ssid);

        startActivity(i);

        finish();

    }

});

}

```

```

voidConvertTexttoSpeech() {

```

```

    ref.addListenerForSingleValueEvent(new ValueEventListener() {

        @Override

        public void onDataChange(DataSnapshot dataSnapshot) {

            distance = dataSnapshot.child(ssid).child("distance").getValue().toString();

        }

    }

```

```

        @Override

        public void onCancelled(DatabaseError databaseError) {

        }

    });

```

```

    String text = "The Obstacle is at " + distance + " centimeters";

```

```

if (distance != null) {

tts.speak(text, TextToSpeech.QUEUE_FLUSH, null);

    }

}


@Override

publicbooleanonOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

caseR.id.About: {

            Intent i = new Intent(getApplicationContext(), About.class);

startActivity(i);

break;

        }

        // case blocks for other MenuItems (if any)

    }

return false;

}

}

```

sregister.xml

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp"
        android:text="Smart Stick"
        android:textAppearance="@style/Base.TextAppearance.AppCompat.Large"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/smart_id"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:layout_marginLeft="10dp"
```

```
android:layout_marginRight="10dp"

android:layout_marginTop="20dp"

android:hint="Enter Smart Stick ID"

android:singleLine="true" />
```

```
<Button
```

```
android:id="@+id/next"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_below="@+id/smart_id"

android:layout_centerHorizontal="true"

android:layout_marginTop="10dp"

android:text="Next" />
```

```
<ImageView
```

```
android:id="@+id/copyright"

android:layout_width="18dp"

android:layout_height="24dp"

android:layout_marginBottom="11dp"

android:layout_alignParentBottom="true"

android:layout_toRightOf="@id/copyright_text"

android:src="@drawable/ic_copyright_black_24dp" />
```

```

<TextView
    android:id="@+id/copyright_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:text="Dept Of CSE"
    android:layout_marginBottom="6dp"
    android:layout_alignBottom="@+id/copyright"
        android:textAppearance="@style/Base.TextAppearance.AppCompat.Large"
    android:layout_alignParentBottom="true"/>

```

```

<android.support.v7.widget.Toolbar
    android:id="@+id/my_toolbar"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="@color/colorPrimary_main"
    android:elevation="4dp"
    android:theme="@style/ThemeOverlay.AppCompat.ActionBar"
    android:titleTextColor="@android:color/white"
    app:popupTheme="@style/ThemeOverlay.AppCompat.Light" />
</RelativeLayout>

```


5. Testing

5.1 Introduction

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product.

It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

5.2 Types of Testing

5.2.1 Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation and user manuals. Organization and preparation of functional tests is focused on requirements, key functions or special test cases.

In addition systematic coverage pertaining to identify Business process flows, data fields, predefined processes and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

5.2.2 Unit Testing

Unit testing is essentially for the verification of the code produced during the coding phase and the goal is test the internal logic of the module/program. In the Generic code project, the unit testing is done during coding phase of data entry forms whether the functions are working properly or not.

In this phase all the drivers are tested they are rightly connected or not. Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two different phases.

5.2.3 System Testing

System Testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results.

An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

5.2.4 Performance Testing

The performance test ensures that the output is produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

5.2.5 Integration Testing

Integration testingis the phase in software testing in which individual software modules are combined and tested as a group. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test. The task of the integration test is to check that components or software applications.

5.2.6 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

5.3 Test Cases

TEST CASE ID	TEST CASE	FUNCTIONALITY	INITIAL STATE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
1	Power ON	Device Start	OFF	Power Supply	Distance Calculation	Distance Calculation	Success
2	Manual mode	Gives distance when user press the button	Do nothing	Not pressed	Do nothing	Do nothing	Success
3	Manual mode activation	Gives distance when user press the button	Do nothing	Press the manual button	Retrieve distance and speak out	Retrieve distance and speak out	Success
4	Auto mode	Moves to automatic mode screen	Do nothing	Click the button	Switches to automatic mode activity	Switches to automatic mode activity	Success
5	Auto mode activation	Retrieve distance in regular intervals of time	Do nothing	Click start button	Speaks distance between obstacle for every 5 seconds	Speaks distance between obstacle for every 5 seconds	Success
6	Auto mode deactivation	Stops automatic retrieval	Speaks distance between obstacle for every 5 seconds	Click stop button	Stops automatic mode	Stops automatic mode	Success

7	Starting Application without internet connectivity	Launch application normally	OFF	Open application	Application opens	Error logging in	Failure
---	--	-----------------------------	-----	------------------	-------------------	------------------	---------

6. Screenshots

SMART STICK

There are two ultrasonic sensors fixed to the stick in such a way that it detects obstacles above the waist as well as below the knee length. The sensors are connected to the raspberry pi which is affixed in a case below the handle. As soon as the user powers up the raspberry pi, the sensors are activated and the distance is calculated continuously.



Figure 6.1: Smart Stick

ULTRASONIC SENSORS

The ultrasonic sensors emit an ultrasound at 40000 Hz which travels through the air and if there is an object on its path, it will bounce back to module. The transmitter transmits short bursts which get reflected by target and are picked up the receiver.

The time difference between transmission and reception between transmission and reception of ultrasonic signals is calculated. Considering the travel time and the speed of the sound you can calculate the distance.

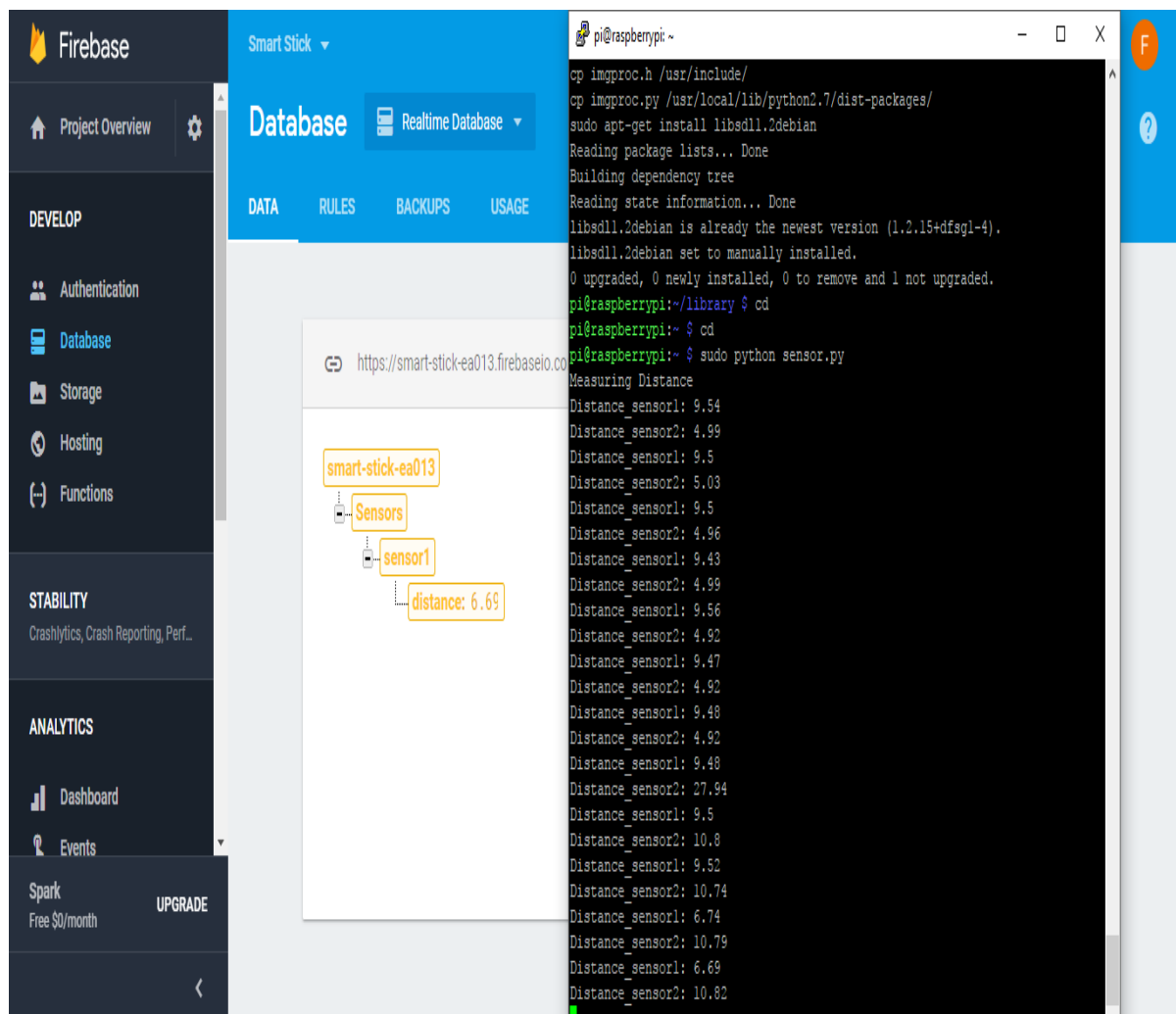


Figure 6.2: Ultrasonic sensors distance

FIREBASE DATABASE

When the distance is calculated from the ultrasonic sensor's pulse signals, the shortest distance among the sensors is sent into the firebase database. This distance is then retrieved by the android application. In Figure.6.2, we can retrieve the value of the distance stored in the database.

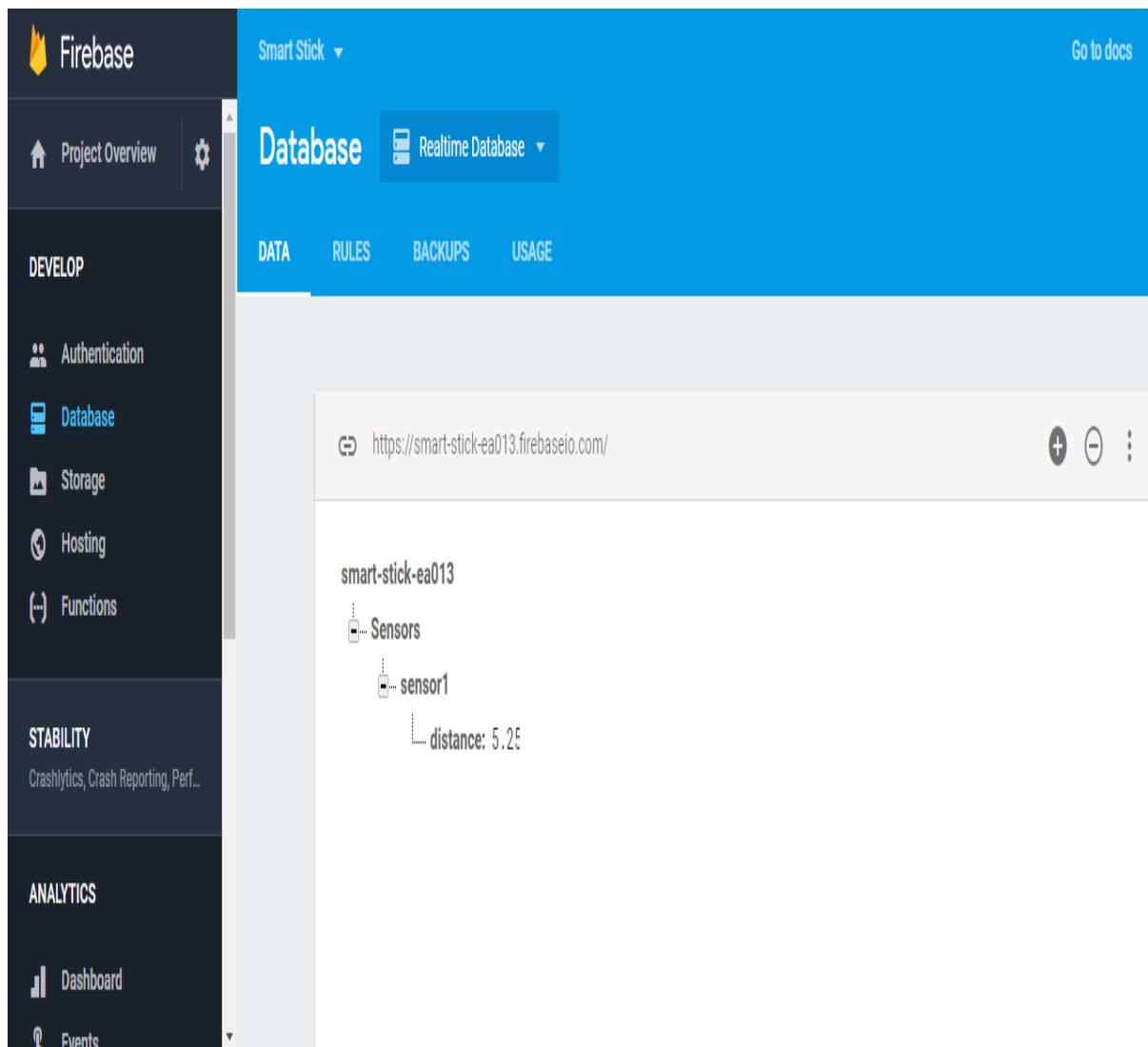


Figure 6.3: Distance value of Smart Stick stored in firebase

SMART STICK APPLICATION

After retrieving the value of distance from the database, it is put into a string which is synthesized for immediate playback. Depending on the mode of operation, the TTS module is run.

In the Manual mode, the TTS module is activated only when the user presses the speak button.

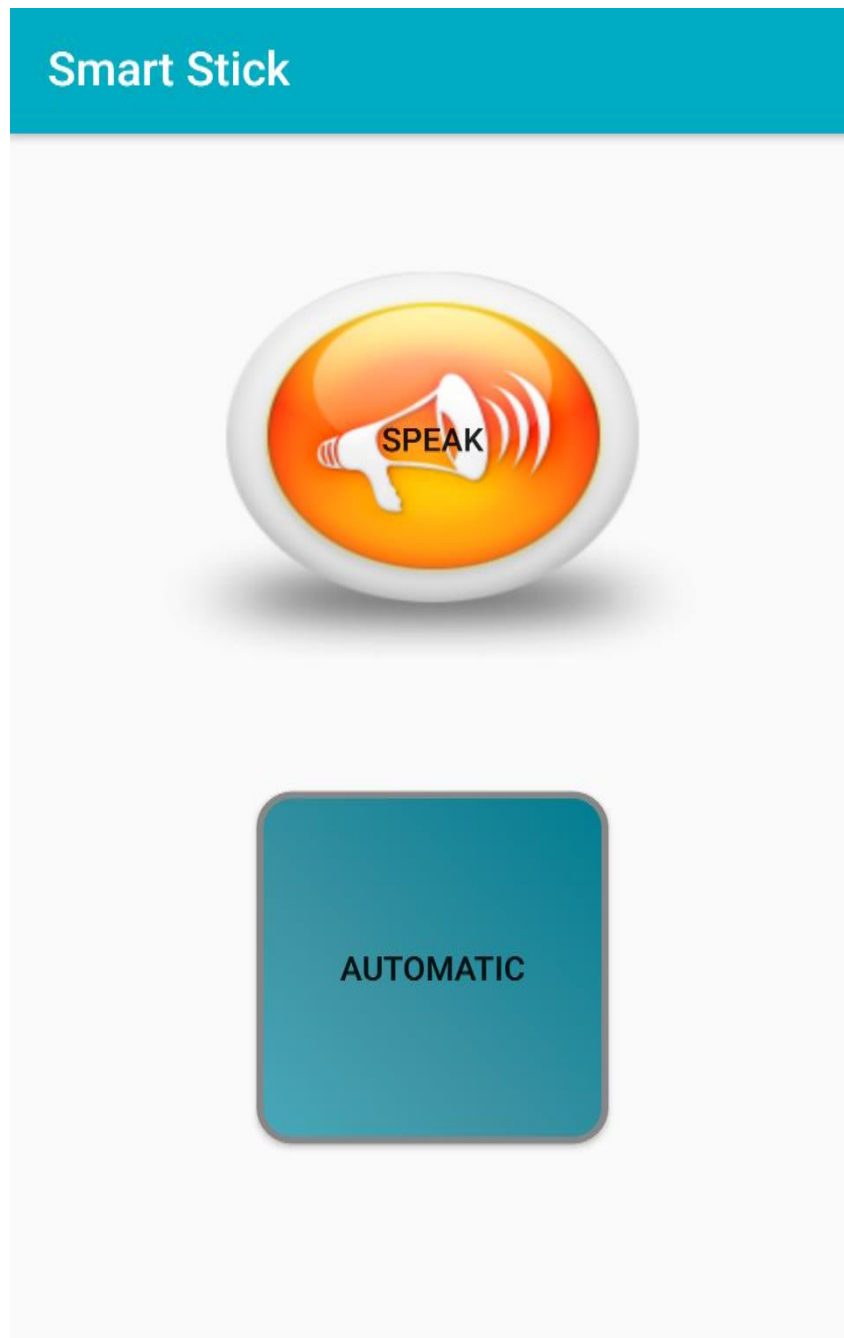


Figure 6.4: Manual Mode (Top Button) of Smart Stick Application

In the automatic mode, the TTS module runs every five seconds.

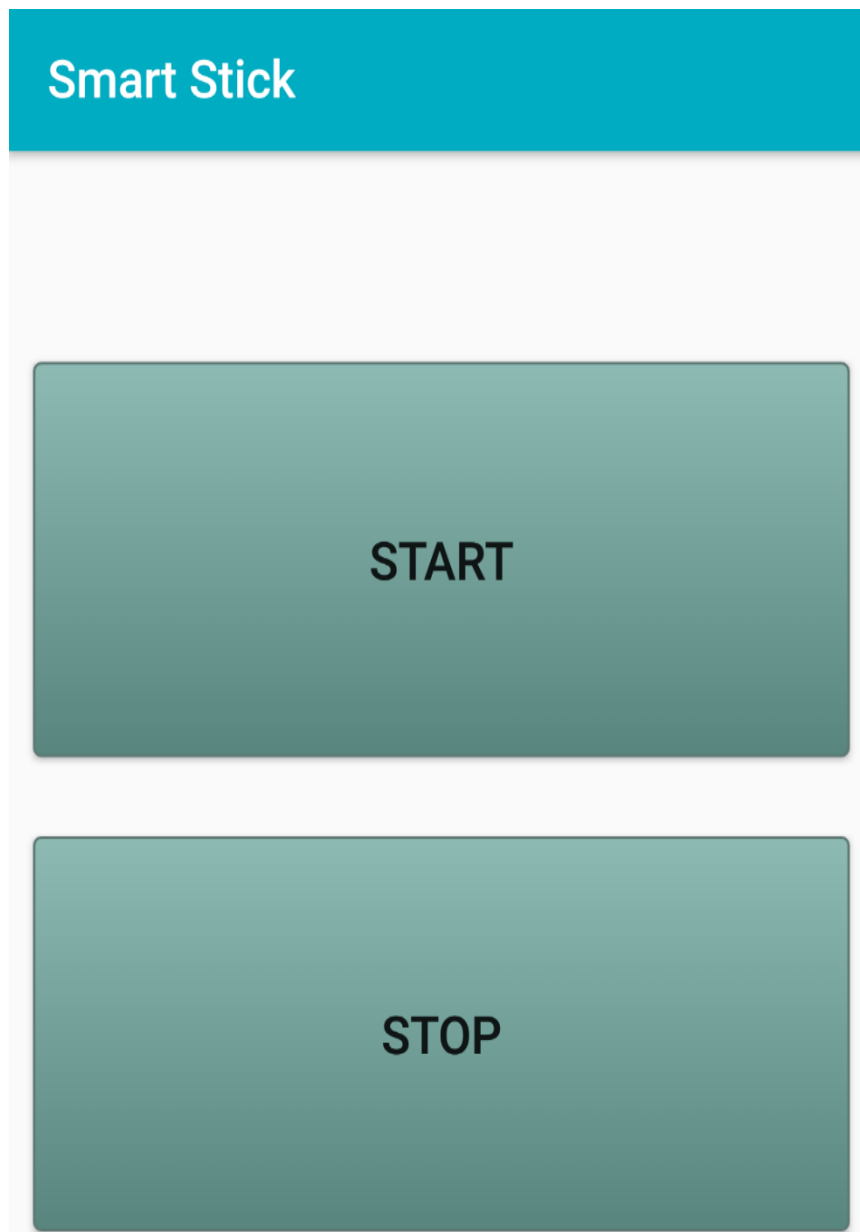


Figure 6.5: Automatic Mode of Smart Stick Application

7. Conclusion

This project is a social welfare project which allows a visually impaired person to travel to any destination with relative ease. Using the Text-to-Speech module, the person can easily know whether the path ahead of them is free or not. It enables the visually impaired person to travel alone without any assistance.

8. Future Enhancements

Enhancements could be done to make the system more mobile as compared to the current design. It can be made more compact so as to make it easier to carry. In addition, if a GPS is installed onto the device, it could also help navigate the person in outdoor environment.

9. Bibliography

References

- [1] <https://www.raspberrypi.org/documentation/>.
- [2] <https://developer.android.com/reference/android/speech/tts/TextToSpeech>.
- [3] AlessioCarullo and MaroParvis, "An ultrasonic sensor for distance measurement in automotive applications," in *IEEE Sensors Journal*, vol. 1, no. 2, pp. 143-, Aug 2001.
- [4] V. Magori, "Ultrasonic sensors in air," *Ultrasonics Symposium, 1994. Proceedings, 1994 IEEE*, Cannes, France, 1994, pp. 471-481 vol.1.
- [5] <https://firebase.google.com/docs/database/>.
- [6] <https://firebase.google.com/docs/android/setup>.
- [7] <https://pythonprogramming.net/gpio-raspberry-pi-tutorials/>.
- [8] AyushWattal, AshutoshOjha and Manoj Kumar, "Obstacle Detection for Visually Impaired Using Raspberry Pi and Ultrasonic Sensors," in National Conference on Product Design (NCPD), July 2016.