

CHAPTER 1

INTRODUCTION

1.1 GENERAL BACKGROUND

The project report provides complete information about the system called “Fire Fighter Robot”. The system is developed as a sensor integrated hardware device for extinguishing fire and a mobile application that helps to control the robot from anywhere. This mobile application also helps to locate the robot. In this section, we are going to give the motivation and overview and the literature Survey.

As the robotic field is developed a lot, human interaction is made less and the robots are widely used for the purpose of safety. Fire accidents have become common in our day-to-day life

And sometimes it may lead to dangerous problems which will be harder for the firemen for protecting the human life. In order to avoid these cases, this robot is used to guard human lives, surroundings and wealth from the fire accidents. The WI-FI technology for remote operation and Raspberry Pi are incorporated in our project. This helps the user to assist him/her while he/she is not around at the home.

Assistive technology is an umbrella term that includes assistive, adaptive, and also includes the process used in selecting, locating, and using them. Fire accidents originate when someone is either sleeping or not at home or due to some carelessness in laboratories, stores etc. By inventing such a device, humans as well as property can be saved at higher rate with minimum damage caused by the fire. As instrumentation engineers, our task was to design and build a prototype system that could autonomously detect and extinguish a fire and also aims at minimizing the air pollution.

1.2. OBJECTIVE

The possibilities of fire are at any remote area or in an industry such as in garments go down, cotton mills, and fuel storage tanks, electric leakages may result in terrible fire & harm. To the worst case of accidents, fire causes heavy loss both financially and by taking lives. These robots are the best possible way, in orders to guard life of humans, surroundings and wealth. It can navigate alone actively and scan the presence of fire and extinguish t. In cases this robot can be used as an emergency device. It is designed in such a manner that could identify the fire as soon as the fire catches and extinguish before the fire spread out and cause heavy damage. The firefighting robot will have future scope that it can work with firefighters, which greatly reduce the danger of injury to victims. It is an innovative work in the field of robotics that operates towards a sensible and obtainable access to save the lives and prevents the danger to property.

CHAPTER 2

LITERATURE SURVEY

2.1 STUDY OF SIMILAR WORKS

Firefighting and rescuing the victims is a risky task. Fire Fighters have to face dangerous situations while extinguishing the fire. Fire Fighters extinguish fires in tall buildings, drag heavy hoses, climb high ladders, and carry victims from one building to another. In addition to long and irregular working hours, fire fighters also face unfriendly environment like high temperature, dust and low humidity. Besides, they also have to face life threatening situations like explosion and collapse buildings. According to the report of IAFF in the year 2000, 1.9 fire fighters per 100,000 structure fires have lost their lives per year in USA. However, this rate was increasing to 3 per 100,000 structure fires. The different causes of Line of Duty Deaths (LODD) are smoke inhalation, burns, crushing injuries and related trauma. Statistics shows that the deaths of fire fighters are constant every year. This results in need of firefighting machines to assist the fire fighters to avoid deaths by handling the dangerous situations. So if a robot is used instead, which can be controlled from a distance or which can perform actions intelligently by itself, which will reduce the risk of this task of firefighting. Robot is a mechanical device that is used for performing tasks that includes high risk like firefighting. There are many types of robots like fixed base robot, mobile robot, underwater robot, humanoid robot, space robot, medicines robot etc. Fixed base robot has limited workspace due to their structure. Workspace of the robot can be increased by using a mobile platform.

Mobile robots can also be used for extinguishing fire in tunnels, industries, hospitals, and laboratory and in homes a firefighting robot will decrease the need of fire fighters to get into dangerous situations. Further the robot will reduce the load of fire fighters. It is impossible to extinguish fire and rescue many victims at a time of huge disaster. Robot technology can be very efficiently used in such cases to rescue much more victims. Thus robotics makes human life easier and safe as well as save a lot of time. The rapid development in technology improves the tools and equipment used in firefighting. These advance tools and equipment can be more effective and efficient. Moreover, it reduces minimum risk level. This will also reduce the damages caused due to a fire incident

2.1.1 EXISTING SYSTEM

The existing fire fighter device helps to pump water when fire is detected. Movement of the robot is controlled based on android application via WI-FI module. It works on the coordinate system or a pattern based system (predefined routes).

2.1.2 DRAWBACKS OF THE EXISTING SYSTEM

- Existing system uses a predefined route, so the robot will only reduce to that predefined location only.
- Existing system cannot be used to locate the robotic vehicle.
- Existing system cannot give any data of fire or temperature status to the android application.
- Existing system cannot be controlled from anywhere.
- Existing system cannot send alerts to the user.

CHAPTER 3

OVERALL DESCRIPTION

The main intention of this project is to design a fire fighting robot using Android application for remote operation. The firefighting robot has a water tanker to pump water and spray it on fire; it is controlled through wireless communication. For the desired operation, Raspberry Pi microcontroller and the Node microcontroller are used.

In the proposed system, an android application is used to send commands from the transmitter end to the receiver end for controlling the movement of the robot in forward, backward, right or left directions. At the receiver side, two motors are interfaced to the Raspberry Pi microcontroller wherein two of them are used for the movement of the vehicle and the Node microcontroller controls the arm of the robot.

Remote operation is done by android OS based Smartphone or tablet. The Android device transmitter acts as a remote control with the advantage of being having adequate range, while the receiver has a Wi-Fi technology for fed to the microcontroller to drive DC motors through the motor driver IC for particular operation. Further, this project is developed by interfacing it with a wireless camera so that the person controlling it can view the operation of the robot remotely on a display.

1.1 PROPOSED SYSTEM

In our proposed system we use android based controlled fire fighter robot that can be used to extinguish fires through remote handling. The vehicle consists of a water tank along with a pump which can throw water when needed. The whole system works based on a Raspberry Pi 3 Model B+.

The android device is used as a transmitter to send over controlling commands to the vehicle, for which an android based application is developed. The android app provides a good touch based GUI for controlling the robot vehicle. Communication between Smartphone and the robot is made possible with the help of Raspberry Pi 3 and the WIFI module. Then the motors are controlled using a driver IC based on the commands received for the vehicle movements in front, back, left and right directions. It allows use of WIFI technology for communication allowing the vehicle to operate in a good range from the device.

This robot can be used to detect fire in disaster prone areas, mainly for apartments, laboratory etc. By means of the GPS module location of the robot is obtained, and also the sensors in the robot used can analysis the temperature level and humidity level and sends the details to the user. The user can control the robot from anywhere of the world using the IOT technology via built in WIFI module. Using this system we can gain access to the exact location by the help of GPS module installed in the android device. We can also identify the fire that may occur in the disaster prone areas with the help of sensors installed in the robot. The robot consist of an automated water pumping arm which is used to extinguish fire at the disaster prone areas.

3.2 FEATURES OF PROPOSED SYSTEM

- The proposed system can control the robot from anywhere using Wi-Fi technology. Doesn't have predefined route.
- Proposed system can locate the robotic vehicle using GPS.
- The proposed system can visualize the objects, in front of the robot to the smartphone using Raspberry Pi camera.
- Proposed system can analysis the room temperature level and humidity level and sends the details to the user.
- Proposed system can send alert notification to the user.
- The proposed system using a robotic arm for pour water to the different directions.

3.3 FUNCTIONS OF PROPOSED SYSTEM

- Efficient
- Cost effective
- User friendly
- Time saving
- Secure
- Remote handling

3.4 REQUIREMENTS OF SPECIFICATION

System analyst talk to a variety of persons to gather details about the business process and their opinions of why things happen as they do and their ideas for changing the process. These can be done through questionnaire', detailed investigation, observation, collection of samples etc. As the details are collected, the analyst study the requirements data to identify features the new system must have, including both the information the system should produce and operational features such as processing controls, response times and input-output methods.

Requirements specification simply means, "Figuring out what is to be made before making it." It determines what people need before starting to develop a product for them. Requirement definition is the activity of translating the information gathered in to a document that defines a set of requirements. These should reflect what consumer wants.

The requirements for an effective fire fighter robot are as follows:

- A real-time monitoring system, to insure the successful extinguishing of fire.
- A system that will work in both daytime and nighttime conditions.
- Additionally, the other features that are implemented on the system must work efficiently.

The above requirements are subsequently the aims of this project. The project will consist of a concept level system that will meet all the above requirements.

3.5 FEASIBILITY ANALYSIS

The initial investigation points to be question whether the project is feasible. The feasibility study concerns with the considerations made to verify whether the system fit to be developed in all terms. Once the idea to develop software is put forward, the question that rises first will pertain to be the feasibility aspects. The prime objective of feasibility study is to ensure that the problem is worth to be solved. At the stage a cost benefit analysis is performed to assertion that the benefit from the system will over rule the cost association with the whole analysis, design and development of the new system. An important outcome of the preliminary investigation determining whether the system required is feasible.

The proposed system is tested in all four aspects of feasibility.

- Technical Feasibility study
- Operational Feasibility study
- Economic Feasibility study
- Behavioural Feasibility study

3.5.1 TECHNICAL FEASIBILITY

The main objective of feasibility study is to test the technical, social and economic feasibility of developing a system. Investing the existing system in the area under investigation and generating ideas about the new system does this. Feasibility study has been done to gather required information. Training, experience and common sense are required for collection of the information. Data was gathered and checked for completeness and accuracy. Analysing the data involved identification of the components of the system and their interrelationship and identified the strength and weakness of the system.

My system is developed by using front end as Python and Android .The front end is technically feasible and it has lot of features as well as its secure too. So the technical part of this project is very secure. So my system is technically feasible.

3.5.2 OPERATIONAL FEASIBILITY

There is no difficulty in implementing the system. The proposed system is effective, user friendly and functionally. The user of the system must be completely unaware of the internal working of the system so that the users will not face any problem running the system. The system thus reduces the responsive time of computer thereby, the system is found to be operationally feasible.

Design is the only ways that can accurately translate the user needs into finished software or system. Without software design, the risk of building an unstable system exists. System design provides the procedural details necessary for implementing the system recommended in the feasibility study.

3.5.3 ECONOMIC FEASIBILITY

Economic and Financial analysis is used for evaluating the effectiveness of the system. The project is technically and operationally feasible.

The software used for developing android application is Android Studio and Sublime. The hardware consists of flame sensors, humidity sensors, Obstacle detector, temperature sensors, an Raspberry Pi 3 Model B+, Raspberry Pi camera, Robotic arm with servos, miniature pump, GPS module, motor driver, relay, robot body, and a water tank. Also a smartphone is needed for controlling and communicate with the robot.

The overall cost for making this prototype robot is considered as Rs.10000/-

3.5.4 BEHAVIOURAL FEASIBILITY

The behavioural feasibility depends upon whether the system performed in the expected way or not. Feasibility study is a test of system proposal according to it workability, impact on organization, ability to meet the user's need and effective use of resources. However, a feasibility study provides a useful starting point for full analysis.

My system is behaviorally feasible because of the effective use of the resources and also the system satisfied the user needs and the system is user friendly.

CHAPTER 4

OPERATING ENVIORNMENT

4.1 HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application.

4.1 Hardware Requirements

- Microcontroller : Raspberry Pi 3 B+ , NodeMCU
 - Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
 - 1GB LPDDR2 SDRAM
 - 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
 - Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
 - Extended 40-pin GPIO header
 - Full-size HDMI
 - 4 USB 2.0 ports
 - CSI camera port for connecting a Raspberry Pi camera
 - DSI display port for connecting a Raspberry Pi touchscreen display
 - 4-pole stereo output and composite video port
 - Micro SD port
 - 5V/2.5A DC power input
 - Power-over-Ethernet (PoE) support
- Motor Driver : L293D Motor Driver
- Gear Motor : 200RPM 12V DC geared motors
- Relay Driver : Single Channel Relay Driver
- Camera : Pi NoIR Camera V2; has a Sony IMX219 8-megapixel sensor.
- Temperature and Humidity Sensor : DHT11
- Water Level and Distance Sensor : HC-SR04-Ultrasonic sensors

- Power Source : Lithium-ion Battery, 15600mAh.
- Robotic Arm with MG90 Servos.
- Flame Sensor.
- Miniature Motor Pump.
- Robot Body.
- Smartphone with Android OS.

4.2 Software Requirements

- Operating System : Windows 10 Pro, Raspbian OS
- Front End : Android Studio IDE, Sublime , Arduino Studio

4.3 TOOLS AND PLATFORMS

4.3.1 Raspberry Pi 3 B+ Microcontroller

The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range, boasting an updated 64-bit quad core processor running at 1.4GHz with built-in metal heatsink, dual-band 2.4GHz and 5GHz wireless LAN, faster (300 mbps) Ethernet, and PoE capability via a separate PoE HAT. With the ARMv8 processor it can run the full range of ARM GNU/Linux distributions, including Snappy Ubuntu Core, as well as Microsoft Windows 10 IoT edition.

Improved thermals on the Pi 3 B+ means that the CPU on the BCM2837 SoC can now run at 1.4GHz, a 17% increase on the previous Pi 3 model (which ran at 1.2GHz). Video performance on Pi 3 B+ is similar to the previous generation Pi 3, the VideoCore being clocked at 400MHz for video processing and the 3D graphics processor running at 300MHz. A significant change on the Pi 3 B+ compared to the Pi 3 is the inclusion of a new faster, dual-band wireless chip (CYW43455) with 802.11 b/g/n/ac wireless LAN and Bluetooth 4.2. The dual-band 2.4GHz and 5GHz wireless LAN enables faster networking with less interference (although the higher bandwidth has less range), and the new PCB antenna technology should allow better reception.

4.3.2 Node MCU

NodeMCU is an open source Lua based firmware for the ESP8266 WiFi SOC from Espressif and uses an on-module flash-based SPIFFS file system. NodeMCU is implemented in C and is layered on the Espressif NON-OS SDK.

The firmware was initially developed as is a companion project to the popular ESP8266-based NodeMCU development modules, but the project is now community-supported, and the firmware can now be run on any ESP module.

4.3.3 L293D Motor Driver

L293D is a typical Motor driver or Motor Driver IC which allows DC motor to drive on either direction. L293D is a 16-pin IC which can control a set of two DC motors simultaneously in any direction. It means that you can control two *DC motor* with a single L293D IC. L293D contains two inbuilt H-bridge driver circuits. In its common mode of operation, two DC motors can be driven simultaneously, both in forward and reverse direction. The motor operations of two motors can be controlled by input logic at pins 2 & 7 and 10 & 15. Input logic 00 or 11 will stop the corresponding motor. Logic 01 and 10 will rotate it in clockwise and anticlockwise directions, respectively.

Enable pins 1 and 9 (corresponding to the two motors) must be high for motors to start operating. When an enable input is high, the associated driver gets enabled. As a result, the outputs become active and work in phase with their inputs. Similarly, when the enable input is low, that driver is disabled, and their outputs are off and in the high-impedance state.

4.3.4 200RPM 12V DC geared motor

200RPM 12V DC geared motors for robotic application. Very easy to use and available in standard size. Nut and threads on the shaft to easily connect and internal threaded shaft for easily connecting it to the wheel.

4.3.5 Relay Driver

Single channel relay driver is an electrically operated device which has a control system and controlled system. A control system is also called as input circuit or input contactor and the controlled system is also called as output circuit or output contactor that are frequently used in automatic control of a circuit. It is an automatic switch that controls a high-current circuit with a low-current signal. Some advantages of a relay are its lower moving inertia, stability, high reliability and small volume. It has wider application in power protection device,

automation technology, sports, remote control, reconnaissance and communication and in electro mechanics and power electronics devices. A relay consist of an induction part that is capable of reflecting the input variable like current, voltage, temperature, pressure, speed, light, power, resistance and frequency etc. To energize or de-energize the connection of controlled circuit an actuator module (output) is present inside it. For coupling and to isolate input current as well as to actuate the output there is an intermediary part between input part and output part is used for the operation to be done. The controlled output circuit of relay will be energized or de-energized when the rated value of input (voltage, current and temperature etc.) is above the critical value.

4.3.6 Camera Module v2 (Pi NoIR)

The infrared Camera Module v2 (Pi NoIR) replaced the original PiNoIR Camera Module in April 2016. The v2 Pi NoIR has a Sony IMX219 8-megapixel sensor (compared to the 5-megapixel OmniVision OV5647 sensor of the original camera).The Pi NoIR gives you everything the regular Camera Module offers, with one difference: it does not employ an infrared filter. (NoIR = No Infrared.) This means that pictures you take by daylight will look decidedly curious, but it gives you the ability to see in the dark with infrared lighting. The camera works with all models of Raspberry Pi 1, 2, and 3. It can be accessed through the MMAL and V4L APIs, and there are numerous third-party libraries built for it, including the Picamera Python library.

4.3.7 Temperature and Humidity Sensor (DHT11)

The DHT11 is a basic, low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). It's fairly simple to use, but requires careful timing to grab data. The only real downside of this sensor is you can only get new data from it once every 2 seconds.

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. Each DHT11 element is strictly calibrated in the laboratory that is extremely accurate on humidity calibration. The calibration coefficients are stored as programmes in the OTP memory, which are used by the sensor's internal signal detecting process. The single-wire

serial interface makes system integration quick and easy. Its small size, low power consumption and up-to-20 meter signal transmission making it the best choice for various applications, including those most demanding ones. The component is 4-pin single row pin package.

4.3.8 HC-SR04-Ultrasonic sensors

HC-SR04 Ultrasonic Sensor is an ultrasonic sensor, also known as an ultrasonic transducer that is based on a transmitter and receiver and mainly used to determine the distance from the target object. The amount of time it takes to send and receive waves will determine how far the object is placed from the sensor. It mainly depends on the sound waves working on “non-contact” technology. The required distance of the target object is measured without any damage, giving you accurate and precise details.

This sensor comes with a range between 2cm to 400cm and is used in a wide range of applications including speed and direction measurement, wireless charging, humidifiers, medical ultrasonography, sonar, burglar alarms, and non-destructive testing.

4.3.9 Lithium-ion batteries

Lithium-ion batteries (LIB) are a family of rechargeable batteries having high energy density and commonly used in consumer electronics. Unlike the disposable lithium primary battery, a LIB uses intercalated lithium compound instead of metallic lithium as its electrode. Usually, LIBs are significantly lighter than other kinds of rechargeable batteries of similar size. LIBs are heavily used in portable electronics. These batteries can be commonly found in PDAs, iPods, cellphones, laptops, etc. This term is also known as a LI-ion.

4.3.10 Flame Sensor

A flame detector is a sensor designed to detect and respond to the presence of a flame or fire, allowing flame detection. Responses to a detected flame depend on the installation, but can include sounding an alarm, deactivating a fuel line (such as a propane or a natural gas line), and activating a fire suppression system. When used in applications such as industrial furnaces, their role is to provide confirmation that the furnace is properly; in these cases they take no direct action beyond notifying the operator or control system. A flame detector can often respond faster and more accurately than a smoke or heat detector due to the mechanisms it uses to detect the flame.

4.3.11 Miniature Motor Pump

The robot consist of miniature water pump which is to pump water from the water tank. It's a 5v DC motor pump. The pump is immersed inside the water. The pump starts working when the command from the relay is obtained when flame is detected.

4.3.12 Robot Body

The robot body consists of wheels which are to drive the robot and extinguishing components like water tank, pump, and sprinkler. A water tank with pump is placed on the robot body and its operation is carried out from the Raspberry Pi o/p through the proper signal from the transmitting end. The entire operation is controlled by an Raspberry Pi. A motor driver IC is interfaced to the Raspberry Pi through which the controller drives the gear motors for the movement of the robotic vehicle.

4.3.13 Android Language

Android is an open source and Linux-based Operating System for mobile devices such as smart phones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies. Android offers a unified approach to application development for mobile devices which means developers need to develop only for Android, and their applications should be able to run on different devices powered by Android. The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007, whereas the first commercial version, Android 1.0, was released in September 2008. On June 27, 2012, at the Google I/O conference, Google announced the next Android version, 4.1 Jelly Bean. Jelly Bean is an incremental update, with the primary aim of improving the user interface, both in terms of functionality and performance. The source code for Android is available under free and open source software licenses. Google publishes most of the code under the Apache License version 2.0 and the rest, Linux kernel changes, under the GNU General Public License version 2.

Android SDK

The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, and Windows 7 or later. As of March 2015, the SDK is not available on Android itself, but software development is possible by using specialized Android applications.

Until around the end of 2014, the officially-supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) Plugin, though IntelliJ IDE (all editions) fully supports Android development out of the box, and NetBeans IDE also supports Android development via a plugin. As of 2015, Android Studio, made by Google and powered by IntelliJ, is the official IDE; however, developers are free to use others, but Google made it clear that ADT was officially deprecated since the end of 2015 to focus on Android Studio as the official Android IDE. Additionally, developers may use any text editor to edit Java and XML files, then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely).

Enhancements to Android's SDK go hand-in-hand with the overall Android platform development. The SDK also supports older versions of the Android platform in case developers wish to target their applications at older devices. Development tools are downloadable components, so after one has downloaded the latest version and platform, older platforms and tools can also be downloaded for compatibility testing. Android applications are packaged in .apk format and stored under data/app/ folder on the Android OS (the folder is accessible only to the root user for security reasons). APK package contains .dex files (compiled byte code files called Dalvik executable), resource files, etc.

4.3.14 Embedded C

Embedded C is a set of language extensions for the C Programming language by the C Standards committee to address commonality issues that exist between C extensions for different embedded systems. Historically, embedded C programming requires nonstandard extensions to the C language in order to support exotic features such as fixed-point arithmetic, multiple distinct memory banks, and basic I/O operations.

In 2008, the C Standards Committee extended the C language to address these issues by providing a common standard for all implementations to adhere to. It includes a number of features not available in normal C, such as, fixed-point arithmetic, named address spaces, and basic I/O hardware addressing. Embedded C uses most of the syntax and semantics of standard C, e.g., main() function, variable definition, data type declaration, conditional statements (if, switch case), loops (while, for), functions, arrays and strings, structures and union, bit operations, macros, etc.

4.3.15 Python

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself. Python is an example of a FLOSS (Free/Libre and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python. Due to its open-source nature, Python has been ported to (i.e. changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.

We can use Python on GNU/Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and PocketPC! We can even use a platform like Kivy to create games for your computer and for iPhone, iPad, and Android. Python supports procedure-oriented programming as well as object-oriented programming. In procedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In object-oriented languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

CHAPTER 5

DESIGN

5.1 SYSTEM DESIGN

System design involves translating system requirements and conceptual design into technical specifications and general flow of processing. After the system requirements have been identified, information has been gathered to verify the problem and after evaluating the existing system, a new system is proposed.

System design is the process of planning of new system or to replace or complement an existing system. It must be thoroughly understood about the old system and determine how computers can be used to make its operations more effective.

There are two levels of system design:

- Logical design.
- Physical design.

In the logical design, the designer produces a specification of the major features of the system which meets the objectives. The delivered product of logical design includes current requirements of the following system components:

- Input design.
- Program design
- Output design
- Database design

Physical design takes this logical design blue print and produces the program software, files and a working system. Design specifications instruct programmers about what the system should do. The programmers in turn write the programs that accept input from users, process data, produce reports and store data in files.

5.1.1 DATA FLOW DIAGRAM/ UML

i. Block Diagram

A block diagram is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. They are heavily used in engineering in hardware design, electronic design, software design, and process flow diagrams. Block diagrams are typically used for higher level, less detailed descriptions that are intended to clarify overall concepts without concern for the details of implementation. Contrast this with the schematic diagrams and layout diagrams used in electrical engineering, which show the implementation details of electrical components and physical construction.

ii. Flowchart

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

iii. Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. A data flow diagram can also be used for the visualization of data processing (Structural design). It is common practice for a designer to draw a context level DFD first which shows the interaction between the system and outside entities the context-level DFD is the "exploded" to show more details of the system being modelled.

The DFD showing the top level of the system is called "Context Diagram". It should be an overview including basic inputs, processes and outputs. Then it is exploded into a more detailed lower level diagram that shows additional features of the system.

The purpose of DFD is to provide a semantic bridge between users and system developers. The diagrams are graphical, eliminating thousands of words, logical representations, modeling what system does; hierarchical, showing system at any level of details; and Jargon less, allowing user interaction and reviewing.

The goal of data flow diagramming is to have a commonly understood model of a system. The diagram is the basis of structured system analysis. The Data flow diagram, also known as “Bubble Chart” has the purpose of clarifying system requirements and identifying major transformations that will become program in system design. The bubble represents the data transformations and the lines represent data flows in the system.

5.1.1.1 BASIC DFD SYMBOLS

- **Rectangles** - representing external entities, which are sources or destinations of data.
- **Arrows** - representing the data flows, this can either be electronic data or physical items. It shows the directional movement of data to and from External Entities, the process and Data Stores.
- **Open-ended rectangles or two parallel lines** – representing data stores, including electronic stores such as databases or XML files and physical stores such as filing cabinets or stacks of paper.
- **Circle or a Rounded Rectangle-** representing processes, which take data as input, do something to it, and output it. It is used to represent functions.

5.1.1.2 COMPONENTS OF DATA FLOW DIAGRAM

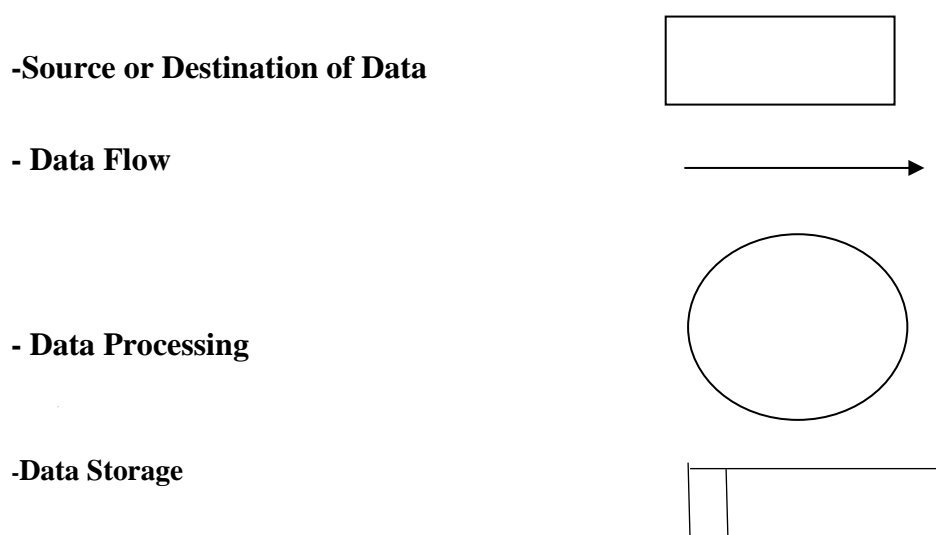


Figure 5.1: DFD components

5.1.2 PROJECT DATA FLOW DIAGRAM/ UML

i. Block Diagram

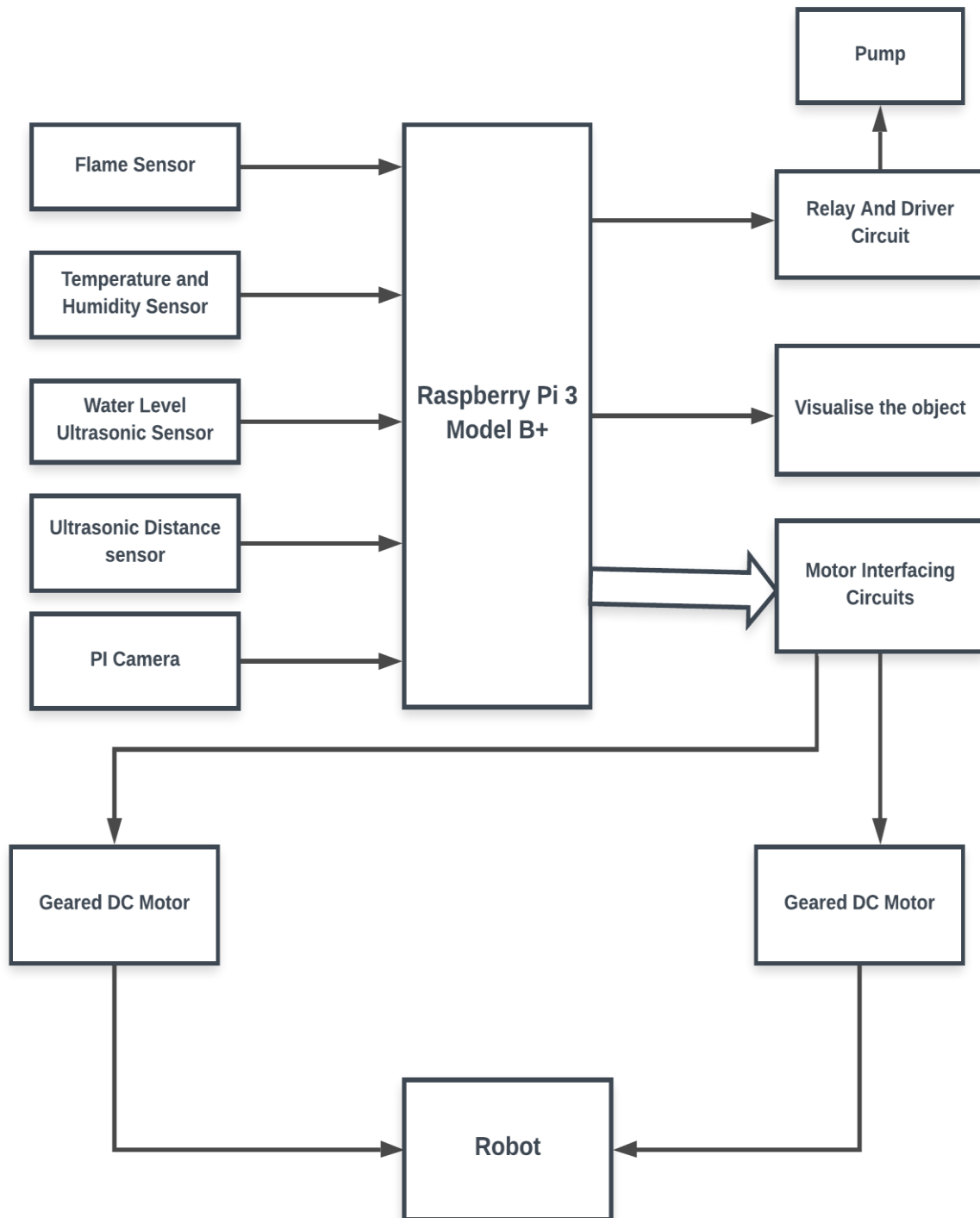


Figure 5.2: Block Diagram

Block Diagram Description

- In this project flame sensors are used to detect the fire and pump the water to the flame.
- Ultrasonic sensors are used to know the water level of the tank also to detect the distance of the vehicle, if the vehicle lost their signal, the vehicle will automatically stop.
- Temperature and humidity sensor is used to detect the room temperature and the humidity level.
- Connect USB Camera to the raspberry pi to know the status of the area and visualize the area in to the screen.
- Motor interfacing circuits helps to controls the robot vehicle to the proper directions.

ii. Use Case Diagram

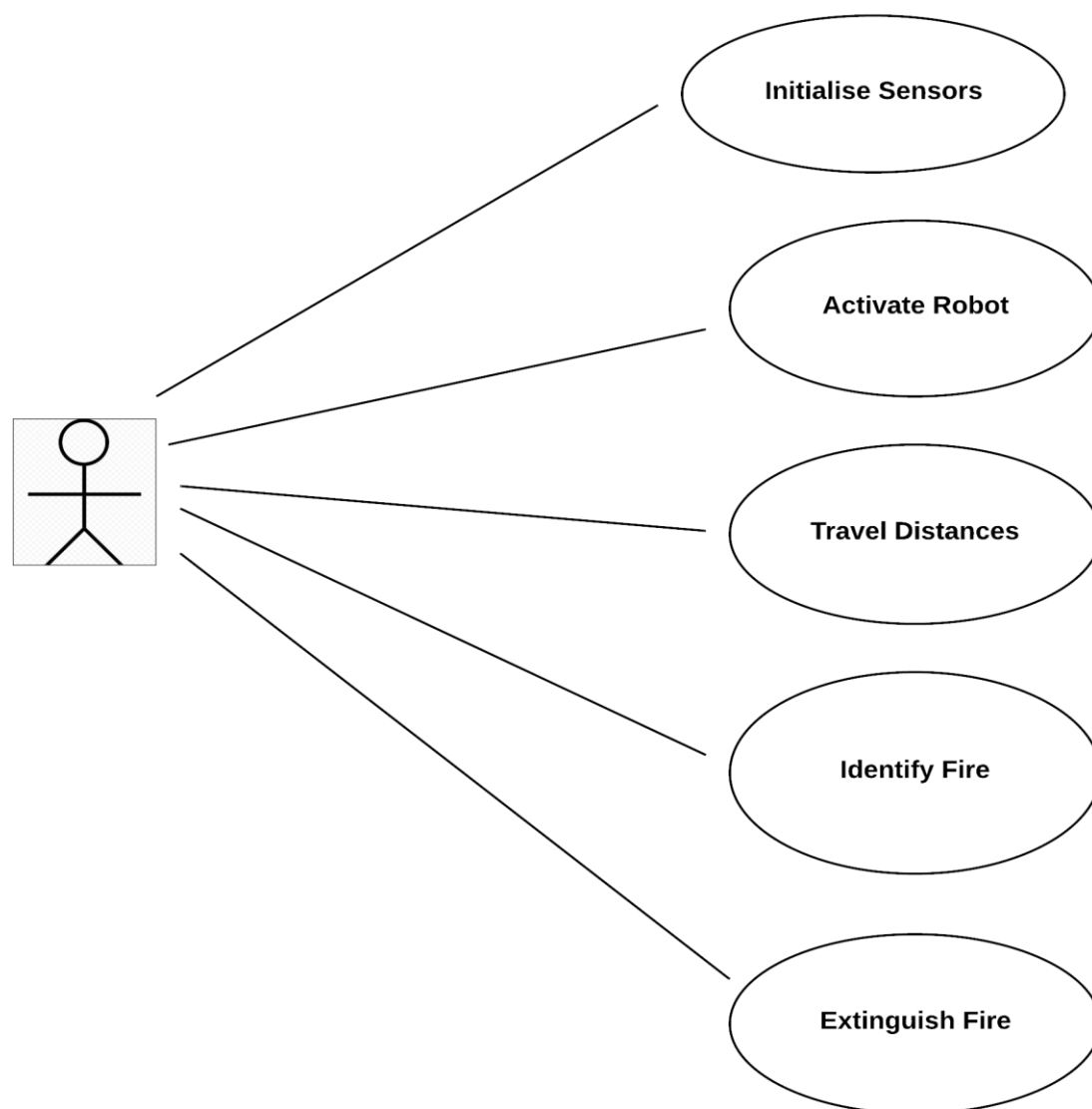


Figure 5.3: Use case diagram

iii. Flow Chart

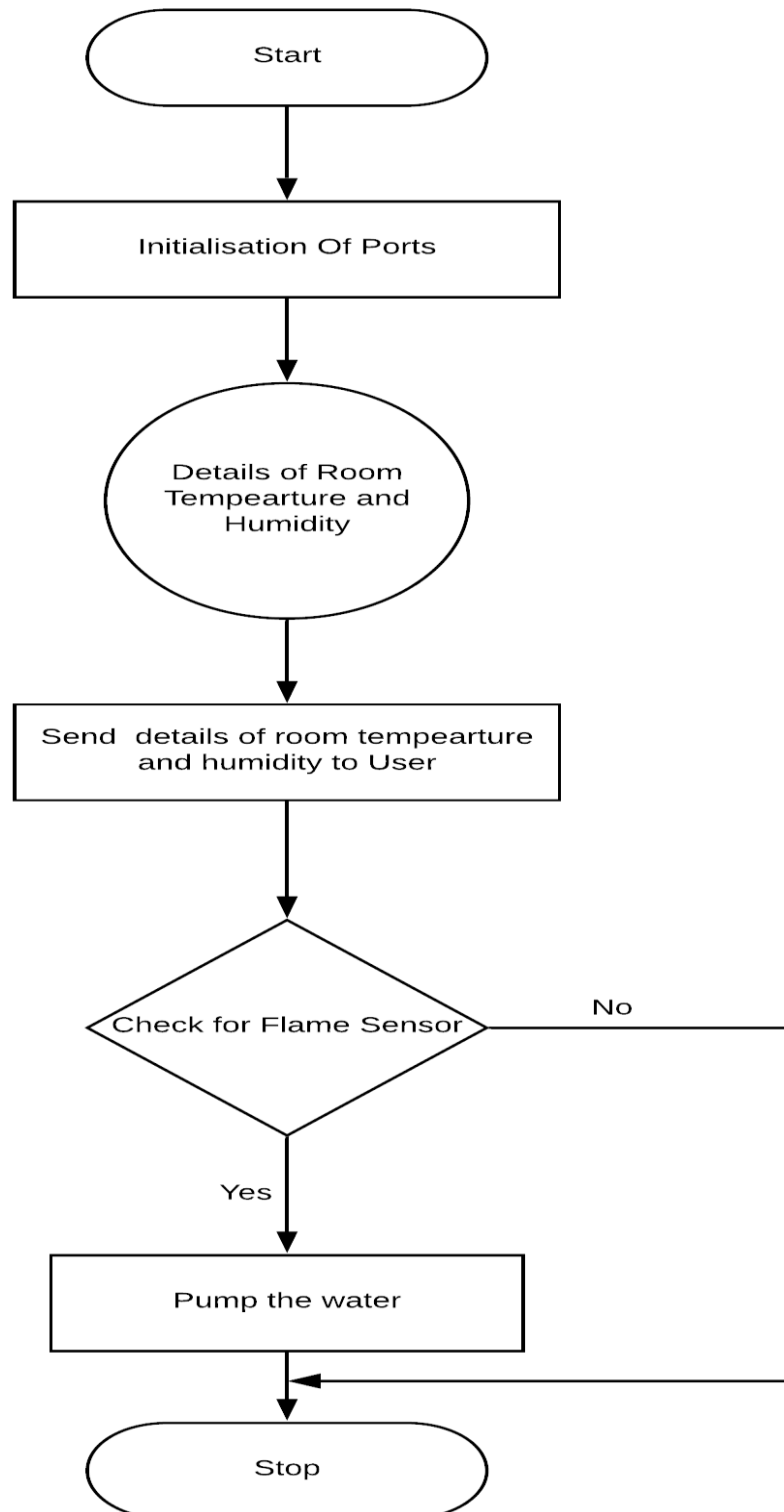


Figure 5.4: Flow Chart

i. Data Flow Diagram

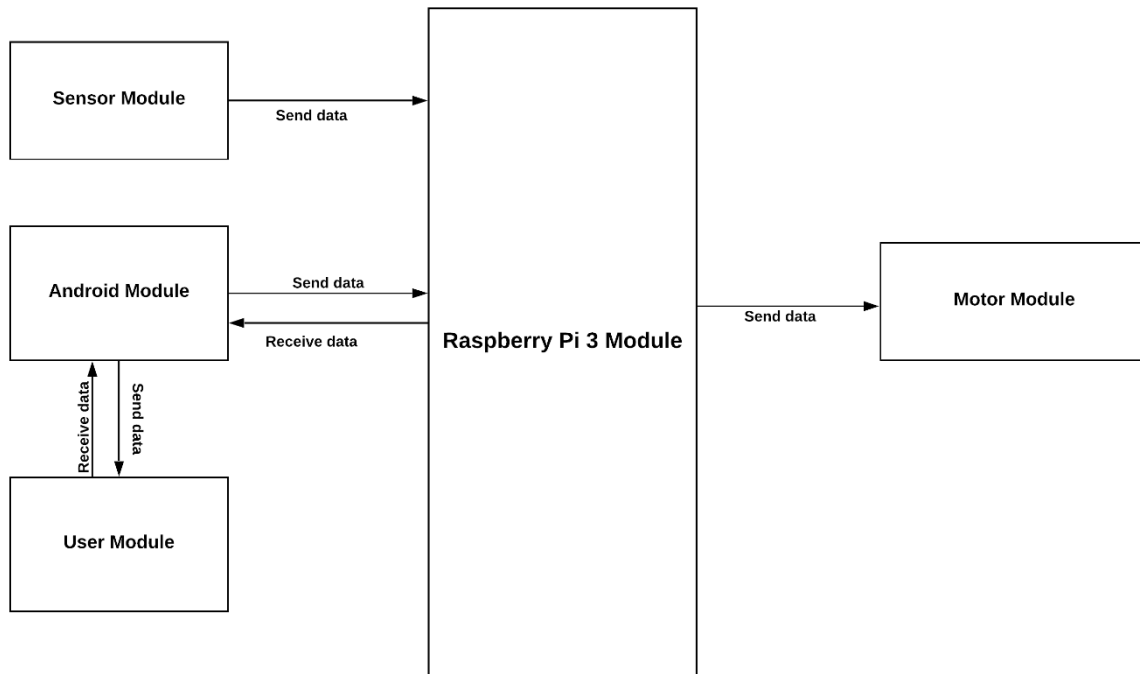


Figure 5.5: DFD

Context Level (Level 0)

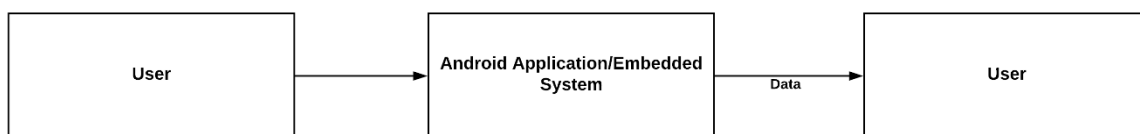


Figure 5.6: Context Level (Level 0)

Level 1

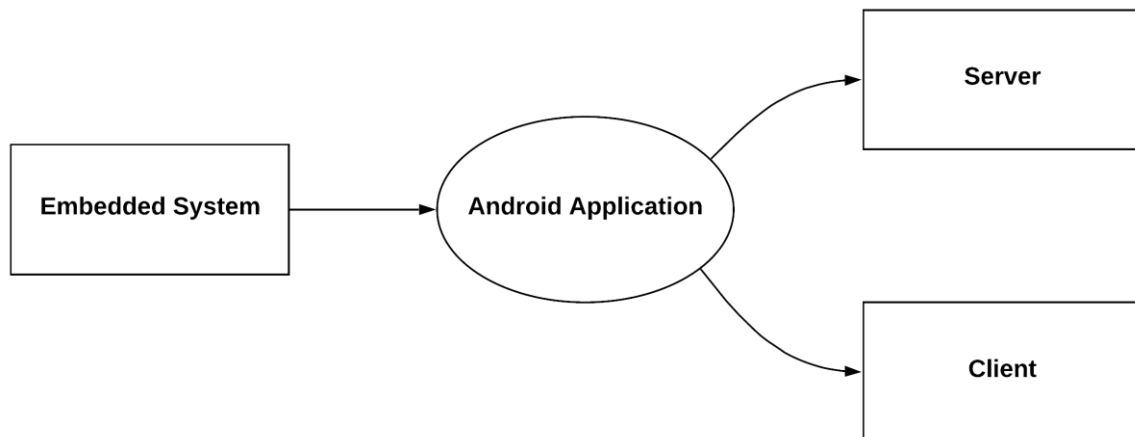


Figure 5.7: Level 1

Level 2

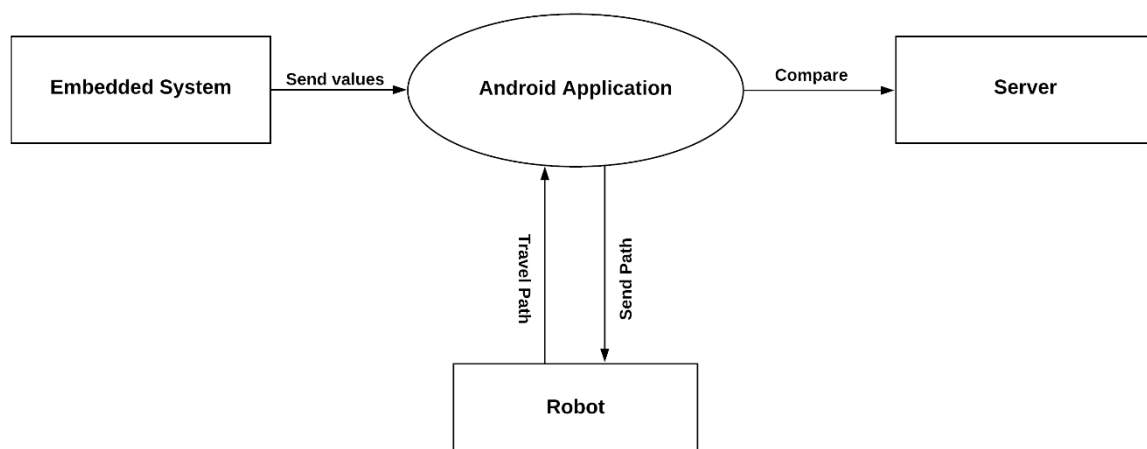


Figure 5.8: Level 2

5.2 DATABASE DESIGN

A database is a collection of interrelated data stored with minimum redundancy to serve users more quickly and efficiently. The general objective of a database is to make information access easy, quick, inexpensive, integrated and shared by different applications and users. Database design is an important yet sometimes overlooked part of the application development lifecycle. An accurate and up-to-date data model can serve as an important reference tool for Database Administrators, developers, and other members of joint application development team. The process of creating a data model helps the team uncover additional questions to ask of end users. Effective database design also allows the team to develop applications that perform well from the beginning. By building quality into the project, the team reduces the overall time it takes to complete the project, which in turn reduces project development costs. The central theme behind database design is to "measure twice, cut once". Effective database designers will keep in mind the principles of normalization while they design a database.

5.3 INPUT DESIGN

Input designing is the basic theory to be considered during system study. The input media used in the system is the keyboard. Details are entered in the system through different data entry screens. The system is designed in a user-friendly manner. Appropriate error messages are displayed when a false data is entered. Design of the system is web-oriented and is highly interactive to the users. The user interface design is very important for any application. The interface design defines how the software communicates within itself, to system that interpreted with it and with human who use it. The interface design is very good; the user will fall into an interactive software application.

The input design is the process of converting the user-oriented description of inputs into a programmer-oriented specification. The objective of input design is to create an input layout that is easy to follow and prevents the user from committing errors. It covers all phases of input, right from the creation of initial databases to the actual data entry into the system. The input design is the link that ties the system into the world of its users. Hence, lays its importance in the design phase. The input design makes sure that while entering data, the end-users understand the format in which the data is to be entered so that it is accepted by the system, the data values that are mandatory for the system to function, the order in which transactions need to be processed etc.

The goal designing input data is to make the automation as easy and free from errors as possible. For providing a good input design for the application easy data input and selection feature and adopted. The input design requirements such as user friendliness, consistent format and interactive dialogue for giving the right message and help for the user at right time are also considered for the development of this project. Input design, involves determining the record media, method of input, speed of capture and entry to the system.

5.4 OUTPUT DESIGN

Output is the most important one to the user. A major form of the output is the display of the information gathered by the system and the servicing the user requests to the system. Output generally refers to the results or information that is generated by the system. It can be in the form of operational documents and reports. Since some of the users of the system may not operate the system, but merely use the output from the system to aid them in decision-making, much importance is given to the output design.

Output generation hence serves two main purposes, providing proper communication of information to the users and providing data in a form suited for permanent storage to be used later on. The output design phase consists of two stages, output definition and output specification. Output definition takes into account the type of outputs, its contents, formats, its frequency and its volume. The output specification describes each type of output in detail.

The objective of the output design to convey the information of all the past activities, current status and emphasize important a quality output is one, which meets the requirements of the end user and presents the information clearly.

Process	Input Design	Output Design
Movement of robot in any direction.	Provides the signal to drive on any direction through the controller on the smartphone.	If the proper signals receives, it moves to correct direction. E.g.: If the signal received as to turn right side, the robot will turn into the right direction.
Visualize the objects in front of the robot.	The raspberry Pi camera that placed in the robot body will take visuals.	It will visualize on the smartphone screen.
Room temperature and humidity.	The temperature and humidity sensor will sense the temperature and the humidity level, and passes the details to user.	The temperature and humidity level are shows in the smartphone screen.
Movement of Robotic arm and pump the water to any direction.	Provides the signal to the servos that are connected in the robotic arm for pumping the water in any direction through the controller on the smartphone.	The water will pump to any direction.
Flame Detection and extinguish the fire.	The flame sensor will sense the fire content.	The robot will automatically pump the water to the fire.
Pump the water if it needed rather than detect fire.	A button is placed on the smartphone screen for pumping the water.	If the signal receives, it pumps the water to anything.

Table 5.1: Input/output Design

5.5 PROGRAM DESIGN

Water level Ultrasonic Sensor

Step 1: If the application is active, the ultrasonic sensor will detect the water level of the water tank.

Step 2: If they need to pour the water, the ultrasonic sensor will receives the information about the water level dynamically.

Step 3: Otherwise, go to step 1.

Distance Ultrasonic Sensor

Step 1: If the application is active, the ultrasonic sensor will detect the distance of the robot from the obstacles.

Step 2: If the obstacles is detected the robot will stops.

Flame Sensor

Step 1: If the application is active, the flame sensor will detect the fire.

Step 2: Then the micro controller receives a signal for pour the water into it.

Step 3: And the information will receives by the micro controller and pump the water to the affected area.

Step 4: Otherwise, go to step1.

Temperature and Humidity Sensor

Step 1: If the application is active, the temperature and humidity sensor will detect the room temperature and the humidity level.

Step 2: Then the received details are shown in the screen.

Camera

Step 1: If the application is active, the camera will visualize the object to the screen.

Step 2: The captured frame will be helped for the hassle free navigation of the vehicle.

CHAPTER 6

FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

6.1 FUNCTIONAL REQUIREMENTS

In software engineering, a functional requirement defines a function of a system or its component. A function is described as a set of inputs, the behavior, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Generally, functional requirements are expressed in the form “system must do requirement”. Functional requirements for each of the uses cases described below:

- Descriptions of data to be entered into the system.
- Descriptions of operations performed by each inputs.
- Descriptions of work-flows performed by the system.
- Descriptions of system outputs.
- How the system meets applicable regulatory requirements.

6.2 NON-FUNCTIONAL REQUIREMENTS

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. Non-functional requirements are “system shall be requirement”. Non-functional requirements are often called qualities of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals", "quality of service requirements" and "non-behavioral requirements. Some of the non-functional requirements are mentioned below:

i. **Performance requirements**

Requirements about resources required, response time, transaction rates, throughput, benchmark specifications or anything else having to do with performance.

ii. **Operating constraints**

List any run-time constraints. This could include system resources, people, needed software, etc.

iii. **Platform constraints**

Discuss the target platform. Be as specific or general as the user requires. If the user doesn't care, there are still platform constraints.

iv. **Accuracy and Precision**

Requirements about the accuracy and precision of the data. Beware of 100% requirements; they often cost too much.

v. **Modifiability**

Requirements about the effort required to make changes in the software. Often, the measurement is personnel effort (person- months).

vi. **Portability**

The effort required to move the software to a different target platform. The

measurement is most commonly person-months or % of modules that need changing.

vii. **Reliability**

Requirements about how often the software fails. The measurement is often expressed in MTBF (mean time between failures). The definition of a failure must be clear. Also, don't confuse reliability with availability which is quite a different kind of requirement. Be sure to specify the consequences of software failure, how to protect from failure, a strategy for error detection, and a strategy for correction.

viii. **Security**

One or more requirements about protection of your system and its data. The measurement can be expressed in a variety of ways (effort, skill level, time) to break into the system. Do not discuss solutions (e.g. passwords) in a requirements document.

ix. **Usability**

Requirements about how difficult it will be to learn and operate the system. The requirements are often expressed in learning time or similar metrics.

CHAPTER 7

TESTING

7.1 TESTING STRATEGIES

An engineered product can be tested in one of the two ways. These testing strategies include:

1 Black box Testing

2 White box Testing

These testing strategy checks the correctness of every statement in the program and results in execution of every instruction in the program module.

1. Black box testing:

Knowing the specified function that a product has been designed to perform, test can be conducted that each function is fully operational. Black box test is carried out to test that input function is properly accepted and output 3 correctly produced. This test examines some aspects of system with little regard for the internal structure of the software.

In our project we use black box testing, we have no clear idea about the coding and we just test the external structure. That is we just try to test that the correct output will come based on the input we give.

Errors found through black box testing are:

- i. Incorrect or missing function.
- ii. Interface errors
- iii. Errors in database structure or external database access.
- iv. Performance errors.

v. Initialization and termination errors.

2. White box testing

White box test of software is predicted on a close examination of procedural detail. The status of the project may be tested at various points to determine whether the expected or asserted status is corresponding to the actual status. We use white box testing, because in this testing all the internal functions and operations are tested. In our project, we have three modules. We separately test these three modules and its internal structure.

Using these following test cases can be derived:

- i. Exercise all logical conditions on their true or false side.
- ii. Exercise all loops within their boundaries and their operation bounds.

7.2 UNIT TESTING

Unit testing focuses on verification effort on the smallest limit of software design. Using the unit test plan prepared in the design phase of the system, important control paths are tested to uncover the errors within the module. This testing was carried out during the coding itself. In this testing step each module is going to be working satisfactorily as the expected output from the module.

Sl.No	Procedures	Expected result	Actual result	Pass or Fail
1	Movement of Vehicle	Vehicle is move to any direction.	Same as Expected	Pass
2	Capture images	Objects are capturing	Same as expected	Pass
3	Movement of arm	Moving of arm to any direction.	Arm can move only 180° direction.	Fail
4	Details of Temperature and Humidity	Detecting the temperature and humidity and shows the details	Same as expected	Pass
5	Information about water level	Detect the water level and shows the details	Same as expected	Pass

Table 7.1: Unit Testing

7.3 INTEGRATION TESTING

It is the systematic technique for constructing the program structure to uncover errors associated with the interface. The objective is to take unit-tested module and built the program structure that has been dictated by design. All modules are combined in this step. Then the entire program is tested as a whole. If a set of errors is encountered connection is difficult because the isolation of causes is complicated by vastness of the entire program. Using this test plan preparing the design phase of the system, the integration was carried out. All the errors found in the system were corrected for the next testing step.

Sl.No	Procedures	Expected result	Actual result	Pass or Fail
1	Pumping of water	Detect flame and automatically pump water in to the flame.	Same as expected	Pass
2	Control the speed of pump	Controlling the speed of water pumping to the area.	Same as expected	Pass

Table 7.2: Integration Testing

7.4 SYSTEM TESTING

In system testing, the software and other system elements are tested as a whole. System testing is actually a series of different test whose primary purpose is to fully exercise the computer-based system.

Sl.No	Procedures	Expected result	Actual result	Pass or Fail
1	Integrate the robot with an android application	Both of them are works perfectly	Same as expected	Pass

Table 7.3: System Testing

7.5 TESTING RESULTS

Sl.No	Test Case	Input	Expected Output	Pass or Fail
1	Application is open	Move to forward and backward	Vehicle moves forward and backward direction	Pass
2	Application is open	Move to left and right	Vehicle moves left and right direction	Pass
3	Movement of arm	Movement to any direction	Check it works on any direction	Fail
4	Movement of arm	Movement to any direction	Check it works on 180° direction	Pass
5	Image Capturing	Capturing the objects	Actual objects	Pass
6	Temperature and humidity detection	Check for temperature and humidity	Temperature and humidity information	Pass
7	Obstacle detection	Check for obstacles	Detect obstacles and stop the vehicle	Pass
8	Obstacle detection	Check for obstacles	Don't detect obstacles	Fail

9	Water level detection	Check water level in the tank.	Provide water level information dynamically	Pass
10	Flame detection	Check for fire	Pour water automatically if fire detected	Pass
11	Flame detection	Check for fire	Not detect the fire	Fail
12	Speed control of water pump	Check the pump power	Control the water force	Pass
13	Finding the vehicle location	Check the location	Provide the exact location of the vehicle	Pass

Table 7.4: Testing results

CHAPTER 8

RESULTS AND DISCUSSION

Results of our experiments show that the robotic vehicle, i.e. transmitter section (android application) and receiver section had been designed and the programs were burned into both the Raspberry Pi 3 B+ and NodeMCU microcontroller. The project is successfully tested for all the commands and it also detected the fire with the help of a flame sensor. Once the flame is detected, a motor drives the water pump. The commands are provided by pressing the switches connected on the transmitter section for forward, backward, right and left movement of the robotic vehicle also the movement of robotic arm.

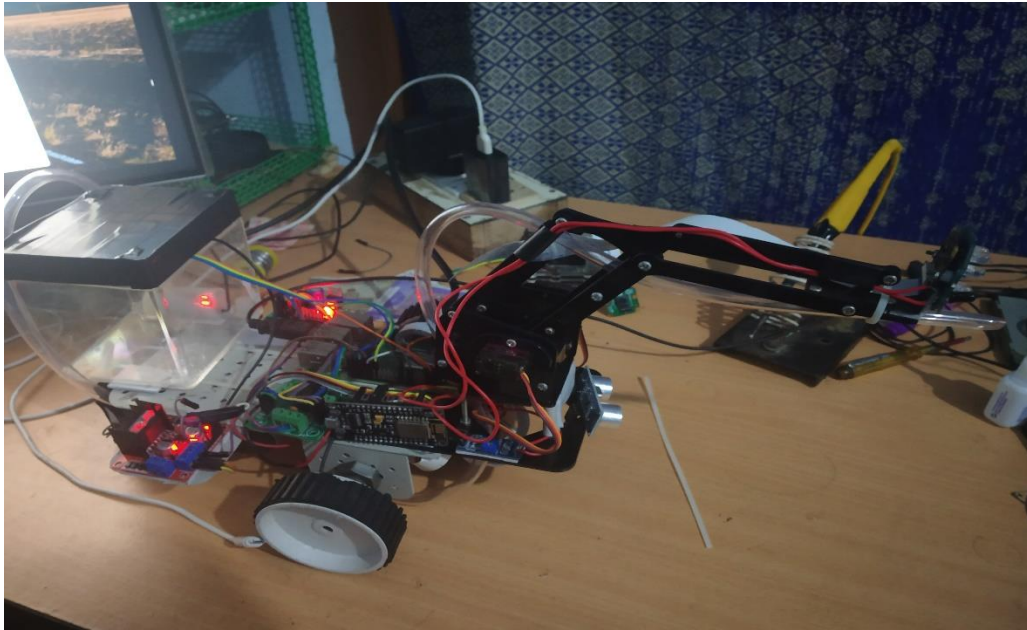
The receiver section also has a GPS module which sends the location of the robot, which was also successfully tested. Main successful part is that, the device can be controlled by the user from anywhere.

8.1 RESULTS

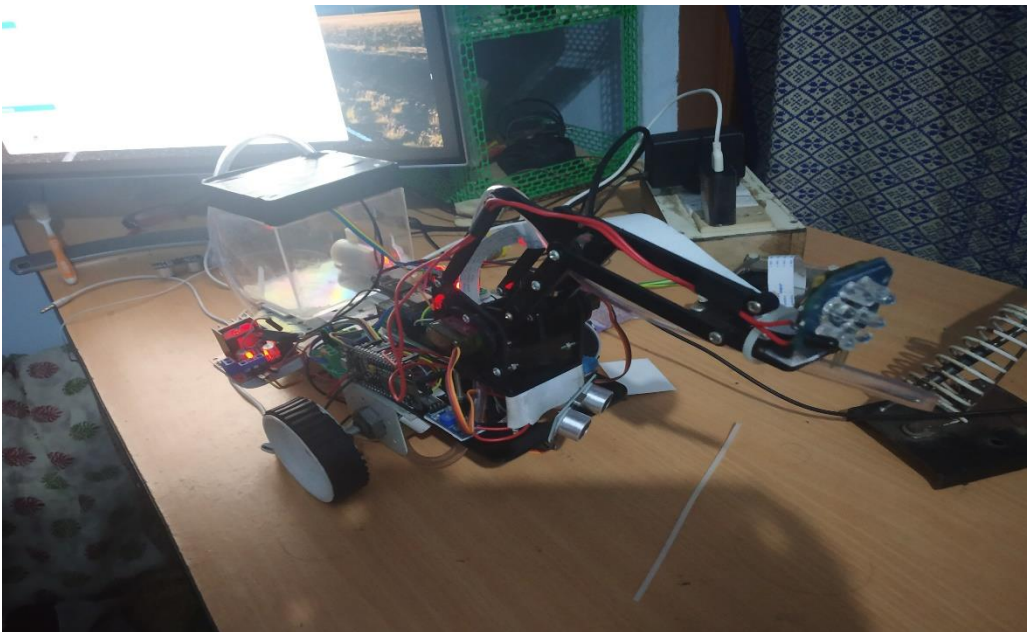
The proposed system incorporated with the following features.

- Quick and appropriate action can be taken easily.
- Human effort can be reduced.
- Improved efficiency.
- Security enhanced for human life.
- Can control remotely from anywhere.
- Give proper information to the user.

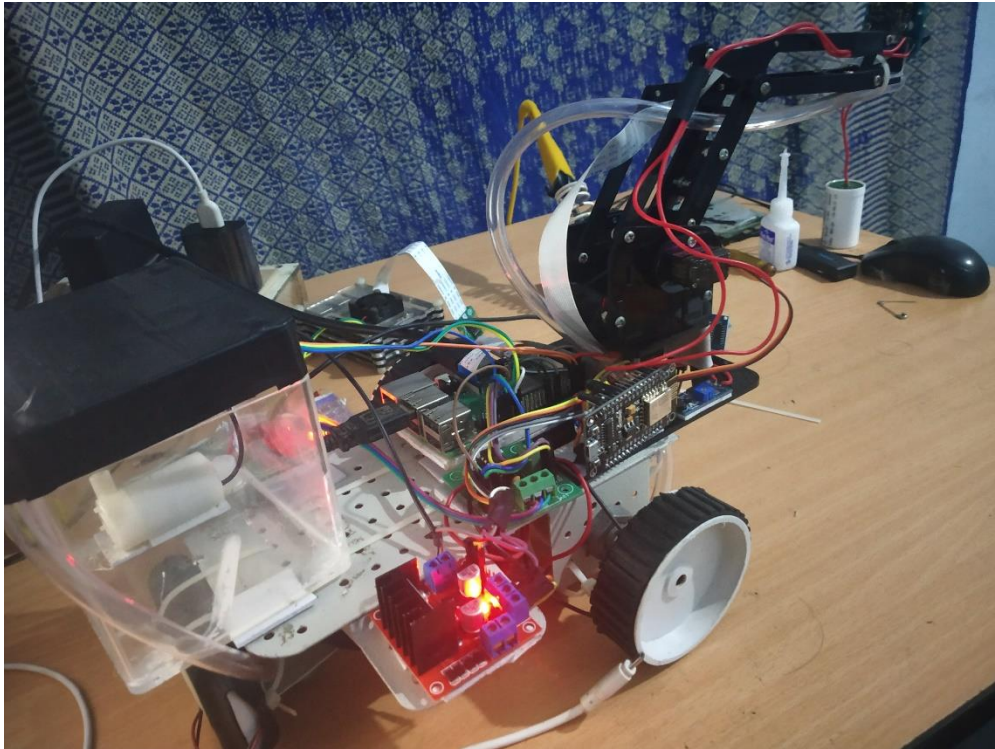
8.2 SCREENSHOTS



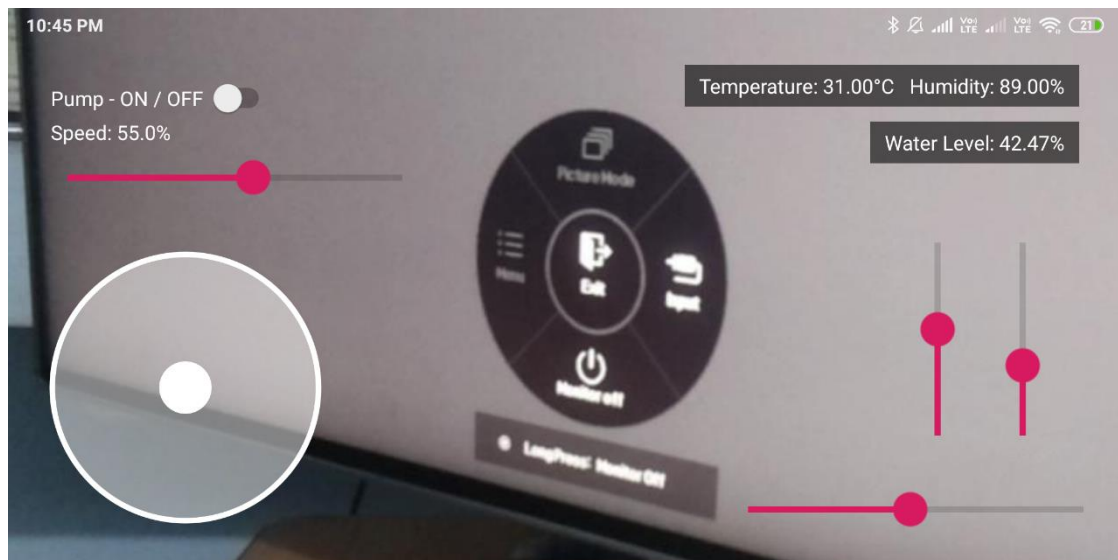
Screen No: 1
Screen Name: Robot vehicle
Description: Side view of the vehicle



Screen No: 2
Screen Name: Robot vehicle
Description: Front view of the vehicle



Screen No: 3
Screen Name: Robot vehicle
Description: Back view of the vehicle



Screen No: 4
Screen Name: camera visual
Description: Visualize the object in front of the camera

CHAPTER 9

CONCLUSION

9.1 SYSTEM IMPLEMENTATION

System Implementation is the stage in the project where the theoretical design is turned into a working system. The implementation phase constructs, installs and operates the new system. The most crucial stage in achieving a new successful system is that it will work efficiently and effectively. The final and important phase in the system in the life cycle is the implementation of the new system.

The term implementation has different meanings ranging from the conversion of a basic application to a complete replacement of a system. The procedure however, is virtually the same. Implementation includes all those activities that take place to convert from old system to new. The new system may be totally new replacing an existing system manual or automated or it may be major modification to an existing system.

This robot is built with an embedded system. It is capable of directing alone on a displayed floor while dynamically scanning the flames of fire. This robot is designed in such a way that it hunts a fire, & soaks it before the fire could spread out of range and control. The robot Consists of temperature sensor, flame sensor, Raspberry Pi 3 B+ NodeMCU, water pump, Ultrasonic distance sensor, L298N motor driver module and geared motor. The robot continuously monitors variation of the surrounding area using these sensors. Robotic vehicle moves through the disaster area as per the instructions from microcontroller. Whenever the flame detected microcontroller identifies that there is presence of fire and operate water pump. Microcontroller operates the relay through the relay interface. Visual indications are attached to the robotic system.

9.2 CONCLUSION

This project (Firefighter Robot) is focused on securing the houses or buildings from fire while the owner is not around. It can be used to extinguish fires through remote handling. The vehicle consists of a water tank along with a pump which can throw water when needed. The whole system works based on a Raspberry Pi 3 B+ and NodeMCU microcontroller. The android device is used as a transmitter to send over controlling commands to the vehicle, for which an android based application is developed. The android app provides a good touch based GUI for controlling the robot vehicle.

Communication between Smartphone and the robot is made possible with the help of Raspberry Pi 3 and the built in WIFI module. By means of the GPS systems, we can obtain the location of the robot exactly by latitude longitude wise and also the sensors in the robot used can analysis the temperature level, humidity level, water level, obstacle detection and sends the details to the user via the android application. The user can control the robot from anywhere in the world using IOT technology via WIFI module, for that an internet connection must be established always.

9.3 FUTURE ENHANCEMENT

The present work can be extended in several ways and some of them are given below:

- For detecting fire with 100% accuracy so that the robot can differentiate between industrial fire and an ordinary flame, we will be adding two more type of sensors i.e. smoke sensor and thermal sensor.
- To save people who get trapped in the fire, we will again use transmission of wireless signals to the fire fighting person so that they can easily locate the people and hence save a lot of precious time.
- We can replace water in pumping system with pressurized carbon dioxide to fight with fires caused due to electric short circuits.
- For domestic use, we will try to implement motion planning using neural networks so that the errors can be minimized in mapping of the house.
- For the better system, we can add Artificial Intelligence with the system, and it gives more efficient and powerful results.

REFERENCE / BIBLIOGRAPHY

1. BOOKS

- i. Danny Staple “Learn Robotics Programming” Build and control autonomous robots using Raspberry Pi 3 and Python, 2017
- ii. Matt Timmons-Brown “Learn Robotics with Raspberry Pi” Build and Code Your Own Moving, Sensing, Thinking Robots, 2015
- iii. Dr Miltiadis A. Bobulous- Automation and Robotics
- iv. A. K. Gupta - Industrial Robotics and Automation

2. WEBSITES

- i. <https://www.python.org/>
- ii. http://www.ijritcc.org/download/conferences/ICMTEST_2016/ICMTEST_2016_Track/1463987827_23-05-2016.pdf
- iii. https://web.iiit.ac.in/~saini_sandeep/report2.doc
- iv. https://www.researchgate.net/publication/269406029_Fire_Fighting_Robot
- v. <http://www.ijecs.in/index.php/ijecs/article/view/3939>
- vi. <https://www.slideshare.net/athmeg/fire-fighting-robot-ppt-56227281>

APPENDICES

1. SCRUM MODEL

i. Git

Git is a version-control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source-code management in software development, but it can be used to keep track of changes in any set of files. As a distributed revision-control system, it is aimed at speed, data integrity, and support for distributed, non-linear workflows.

ii. Git Repositories

A Git repository contains the history of a collection of files starting from a certain directory. The process of copying an existing Git repository via the Git tooling is called cloning. After cloning a repository the user has the complete repository with its history on his local machine. Of course, Git also supports the creation of new repositories. If you want to delete a Git repository, you can simply delete the folder which contains the repository. If you clone a Git repository, by default, Git assumes that you want to work in this repository as a user.

iii. Scrum

Scrum is an agile way to manage a project, usually software development. Agile software development with Scrum is often perceived as a methodology; but rather than viewing Scrum as methodology, think of it as a framework for managing a process. In the agile Scrum world, instead of providing complete, detailed descriptions of how everything is to be done on a project, much of it is left up to the Scrum software development team.

In the agile Scrum world, instead of providing complete, detailed descriptions of how everything is to be done on a project, much of it is left up to the Scrum software development team. Within agile development, Scrum teams are supported by two specific roles. The first is a Scrum Master, who can be thought of as a coach for the team, helping team members use the Scrum process to perform at the highest level. The product owner (PO) is the other role, and in Scrum software development, represents the business, customers or users, and guides the team toward building the right product.

iv. Git History


9 commits






1 branch


0 releases

1 contributor

Branch: master [New pull request](#) [Find file](#) [Clone or download](#)

 muneersha initial commit Latest commit 56fr755 10 days ago

 Robotic-Arm-Control-ESP32-master/roboticArm	arm control	10 days ago
 Robotic-Fire-Engine-Android-App-master	initial commit	10 days ago
 Robotic-Fire-Engine-Python-master	python	29 days ago
 .gitattributes	Initial commit	29 days ago
 README.md	Update README.md	29 days ago

 README.md

FIRE FIGHTER


The proposed project aims to develop an android controlled fire fighter robot that can be used to extinguish fires through remote handling. The vehicle consists of a water tank along with a pump which can throw water when needed. The whole system works on a Raspberry Pi 3 Model B+. The android device is used as a transmitter to send over controlling commands to the vehicle, for which an android based application is developed. The android app provides a good touch based GUI for controlling the robot vehicle. Communication between Smartphone and the robot is made possible with the help of Raspberry Pi 3 and the built in Wi-Fi module on the Raspberry Pi 3. Then the motors are controlled using a driver IC based on the commands received for the vehicle movements in front, back, left and right directions. It uses WI-FI technology for communication, allowing the vehicle to operate from anywhere in the world based on IOT. There is a GPS module to obtain

muneersha / FIRE-FIGHTER











[Watch](#) 0 [Star](#) 0 [Fork](#) 0


[Code](#) [Issues](#) 0 [Pull requests](#) 0 [Projects](#) 0 [Security](#) [Insights](#)

Branch: master [FIRE-FIGHTER / Robotic-Fire-Engine-Python-master /](#) [Create new file](#) [Find file](#) [History](#)

 muneersha python Latest commit be16efd 29 days ago

..

 build	python	29 days ago
 node_modules	python	29 days ago
 README.md	APPLICATION	29 days ago
 dump.txt	python	29 days ago
 dump.txt~	python	29 days ago
 index.html	python	29 days ago
 index.html~	python	29 days ago
 package.json	python	29 days ago
 server.js	python	29 days ago
 server.js~	python	29 days ago

 README.md

muneersha / FIRE-FIGHTER
 Watch 0 Star 0 Fork 0

<> Code
 Issues 0
 Pull requests 0
 Projects 0
 Wiki
 Security
 Insights
 Settings

Branch: master
 FIRE-FIGHTER / Robotic-Fire-Engine-Android-App-master /
 Create new file
 Upload files
 Find file
 History

muneersha initial commit
 Latest commit 56fc755 10 days ago

..		
.idea	APPLICATION	29 days ago
app	initial commit	10 days ago
gradle/wrapper	APPLICATION	29 days ago
.gitignore	APPLICATION	29 days ago
README.md	APPLICATION	29 days ago
build.gradle	APPLICATION	29 days ago
gradle.properties	APPLICATION	29 days ago
gradlew	APPLICATION	29 days ago
gradlew.bat	APPLICATION	29 days ago
settings.gradle	APPLICATION	29 days ago

README.md

muneersha / FIRE-FIGHTER
 Watch 0 Star 0 Fork 0

<> Code
 Issues 0
 Pull requests 0
 Projects 0
 Wiki
 Security
 Insights
 Settings

History for FIRE-FIGHTER / Robotic-Arm-Control-ESP32-master / roboticArm / roboticArm.ino

Commits on May 22, 2019

arm control
 muneersha committed 10 days ago

d0bbefa
 <>

[muneersha / FIRE-FIGHTER](#)

Watch
 0

Star
 0

Fork
 0

[Code](#)
[Issues 0](#)
[Pull requests 0](#)
[Projects 0](#)
[Security](#)
[Insights](#)

History for [FIRE-FIGHTER](#) / Robotic-Fire-Engine-Android-App-master

Commits on May 22, 2019

initial commit

muneersha committed 10 days ago

56fc755

Implemented speed control

muneersha committed 10 days ago

280386d

Implemented robotic arm control functionality

muneersha committed 10 days ago

a11d027

Commits on May 4, 2019

APPLICATION

muneersha committed 29 days ago

06b5c6d

[muneersha / FIRE-FIGHTER](#)

Watch
 0

Star
 0

Fork
 0

[Code](#)
[Issues 0](#)
[Pull requests 0](#)
[Projects 0](#)
[Security](#)
[Insights](#)

History for [FIRE-FIGHTER](#) / Robotic-Fire-Engine-Python-master

Commits on May 4, 2019

python

muneersha committed 29 days ago

be16efd

APPLICATION

muneersha committed 29 days ago

06b5c6d

2. LIST OF TABLES

Table No	Table Name	Page No
5.1	Input output design	28
7.1	Unit test cases and results	35
7.2	Integration test cases and results	36
7.3	System test case and results	36
7.4	Testing result	37-38

3. LIST OF FIGURES

Figure No	Name	Page No
5.1	DFD Components	20
5.2	Block diagram	21
5.3	Use case diagram	22
5.4	Flow chart	23
5.5	Data Flow Diagram	24
5.6	DFD Context level(Level 0)	24

5.7	DFD Level 1	25
5.8	DFD Level 2	25

4. ABBREVIATIONS AND NOTATION

i. DFD

DFD (Data Flow Diagram) is a graphical representation of the "flow" of data through an information system, modeling its process aspects.

ii. GPS

GPS (Global Positioning System) is a satellite-based radio navigation system.

iii. MCU

MCU (Micro Controller Unit) is a small computer on a single integrated circuit. A microcontroller contains one or more processor cores along with memory and programmable input/output peripherals.

iv. UML

UML (Unified Modelling Language) is a general purpose modelling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering.

5. CODING

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

<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.muneer.fireengine">

    <uses-permission
android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"

        android:networkSecurityConfig="@xml/network_security_config"
    >

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

```

</manifest>

{
  "name": "FireEngine",
  "version": "0.0.0",
  "description": "",
  "main": "server.js",
  "dependencies": {
    "express": "^4.16.4",
    "mmm-sonic": "^2.5.3",
    "socket.io": "^2.2.0"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit
1"
  },
  "author": "Muneersha BL",
  "license": "ISC"
}

```

```

var app =
require('express')();

var http = require('http').Server(app);
var io = require('socket.io')(http);

var gpio = require('rpi-gpio')
var gpiop = gpio.promise;
var gpio1 = require("gpio");
const Gpio = require('pigpio').Gpio;
// public directory
app.get('/', function(req, res){

```

```

    res.sendFile(__dirname + '/index.html');
  });
  // Motor pin
  const motorPump = 7;
  // ultrasonic sensor
  const MICROSECDONDS_PER_CM = 1e6/34321;
  const trigger = new Gpio(5, {mode: Gpio.OUTPUT});
  const echo = new Gpio(6, {mode: Gpio.INPUT, alert: true});
  trigger.digitalWrite(0);
  let totalTankCap = 15;
  let tankOffsetDist = 3;
  const watchHCSR04 = () => {
    let startTick;
    echo.on('alert', (level, tick) => {
      if (level == 1) {
        startTick = tick;
      } else {
        const endTick = tick;
        const diff = (endTick >> 0) - (startTick >> 0);
        var dist = diff / 2 / MICROSECDONDS_PER_CM;
        let tankDist = dist - tankOffsetDist;
        let level = totalTankCap - tankDist;
        let perc = level / totalTankCap * 100;
        if(perc >= 100)
          perc = 100;
      }
    });
  };

  levelJson = {'water': perc.toFixed(2)};
  io.emit('water', levelJson);

```

```

    });});
watchHCSR04();
// humidity and temperature sensor
var rpiDhtSensor = require('rpi-dht-sensor');

var dht = new rpiDhtSensor.DHT11(2);
function read () {
    var readout = dht.read();
    item = {'temp': readout.temperature.toFixed(2), 'humidity':
readout.humidity.toFixed(2)};
    console.log(item);
    return item;
}

```