INTERNET OF THINGS BASED MECHATRONIC SYSTEM

David Simpson

Bachelor of Engineering
Mechatronic Engineering

Department of Electronic Engineering
Macquarie University

November, 2017

Supervisor: Prof. Subhas Mukhopadhyay

ACKNOWLEDGMENTS

I would like to acknowledge the people who made it possible for me to make it this far in my academic career. To my parents, thank you for the unconditional love and support you have shown me as I struggled to find my own path. Without all you have done for me it is doubtful I would have made it this far. To my partner Kim, thank you for all the love you show me and for keeping me sane over these years of study. To my brother, thank you for showing me how to overcome my problems and the rewards of hard work. Finally I would like to thank Prof. Subhas Mukhopadhyay who has provided me with invaluable personal and academic advice throughout this project. Thank you for the kindness and flexibility you have shown me and for your friendly and approachable attitude.

STATEMENT OF CANDIDATE

I, David Simpson, declare that this report, submitted as part of the requirement for the award of Bachelor of Engineering in the Department of Electronic Engineering, Macquarie University, is entirely my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualification or assessment an any academic institution.

Student's Name: David Simpson

Student's Signature: David Simpson (Electronic)

Date: 13/11/2017

ABSTRACT

This thesis has the purpose of developing a new Internet of Things controlled mechatronic system. The ability to remotely control mechatronic systems opens up a world of possibilities for new devices as well as upgrading existing technologies. Throughout this project a new Internet of Things framework was created from scratch including an Android application, cloud based web server and a mobile mechatronic system to execute user commands. All of these project components were newly developed and designed to provide a potential teaching platform for future students of wireless mechatronics. The Android application communicates user commands to the web server which acts as a 'fixed point' to which both the mechatronic system and the Android application can communicate. The web server saves any received command data into an SQL database for analysis and then encrypts the data. The mechatronic system is a mobile robot designed from the ground up using CAD tools and then 3D printed to allow for future design changes. The robot polls the web server at a set interval to check for new commands. The overall Internet of Things framework developed during this project is functional but requires further software development to make the system more robust and completely secure.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Throughout history humanity has understood the power and importance of communicating over large distances and has strived to develop new and improved methods to accomplish this goal. One of the earliest rudimentary methods was the use of smoke signals to indicate danger [4,13,15] or to send news, for example the Chinese use of signal beacons to alert their troops to the presence of approaching enemies. These smoke signals provided an invaluable advantage allowing for troops to be readied and positioned rather than caught off guard. Further early communication methods that have been used to great effect include homing pigeons, signal drums and semaphore systems [4,15]. Each of these techniques provided their user with a strong advantage over their competitors, be it in war or business.

After the discovery of electricity the true power of long distance communication began to be unlocked. With the invention of the electrical telegraph system suddenly individuals, businesses and governments could send messages over huge distances almost instantaneously, resulting in massive economic and social impacts. In a relatively short time humanity went on to develop the telephone, as well as one of the first wireless electrical communication tools, the radio. Over the course of the 20th century microwave and satellite communication systems were developed along with the first mobile phones and the modern internet.

The Internet of Things (IoT) is the next step forward on this developmental path. IoT is a broadly defined concept relating to the trend of increased connectivity of modern products and devices. While there is no official consensus of what constitutes the IoT there are two definitions that I believe provide insight into the topic:

- The Oxford English Dictionary: "*The interconnection via the Internet of computing devices embedded in everyday objects, enabling them to send and receive data [20].*"
- The Internet Architecture Board: "*The term Internet of Things (IoT) denotes a trend where a large number of embedded devices employ communication services offered by the Internet protocols. Many of these devices, often called smart objects, are not directly operated by humans, but exist as components in buildings or vehicles, or are spread out in the environment [20].*"

1.1 Project Goals and Deliverables

The over-arching goal is to create an IoT based mechatronic system. It is intended that the final deliverables of this project include an Android smartphone control app, an arduino based robot and a web server to facilitate communication between the app and robot while recording important data for analysis. It is my hope that the outcomes of this project can find a potential use as a teaching platform for future students of wireless mechatronics. A list of project deliverables is provided below:

- Android smart-phone control application.
- Cloud-based web-server to facilitate communication.
- SQL database for data analytics.
- Prototype mobile robotic platform with attached robotic arm.
- Several additional robotic platforms based on prototype.
- Collision avoidance behaviour implemented.

Chapter 2

Background and Related Information

2.1 The Internet of Things

The term ”Internet of Things” was first used by researcher Kevin Ashton in 1999 during a presentation where he discussed expanding the internet to accommodate connected objects or ’Things’ [17, 20]. As per the definitions given in Chapter 1, the IoT can be loosely defined as the interconnection and communication between distributed products, objects and devices with the goal of providing improved services and performance through convenient user access and data analysis. An example to clarify this definition would be a ’smart’ house where distributed sensor nodes constantly measure temperature and communicate with the air conditioning control system to regulate the temperature as required. The home-owner would be able to access these temperature readings and control the air-conditioning system remotely over the internet. The household power usage could also be monitored by a smart-meter allowing for a home-owner to examine their energy usage and take action to reduce their power bills. This is just a basic example of the potential value provided by IoT connected devices. This value will continue to grow as the number of connected ’things’ rapidly expands. This forecast future growth is shown in Figure 2.1.

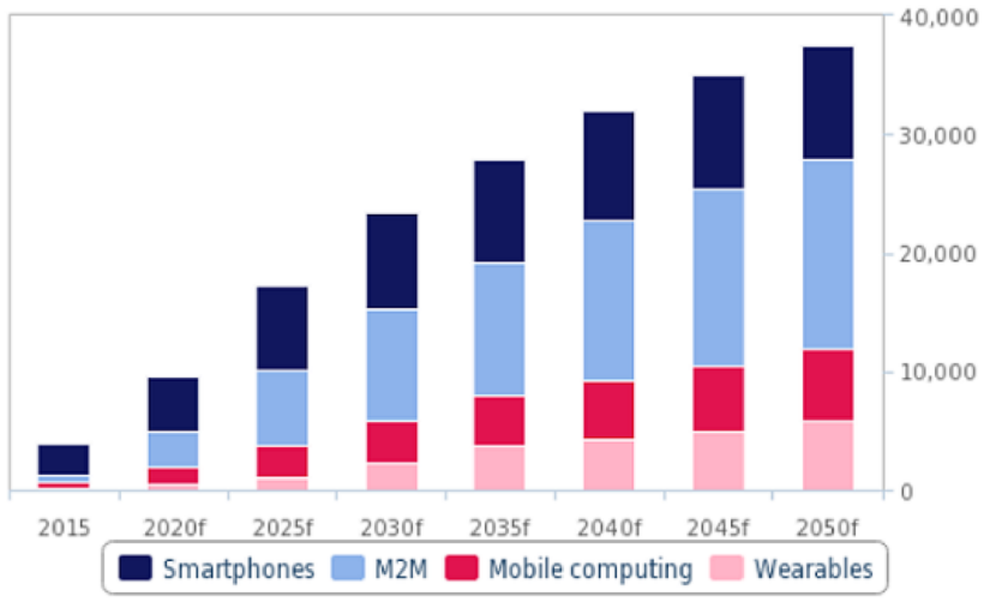


Figure 2.1: Forecast number of connected devices (millions). [11]

IoT System Architecture

The 'smart' house scenario given in Section 2.1 provides a common example of an IoT home automation system. The next step in understanding IoT systems is to look at a basic system architecture model that could be implemented for a variety of IoT applications. Figure 2.2 provides the outline for a simplified IoT system. There are limitless potential applications for the IoT and so to help define the different types of systems the Internet Architecture Board (IAB) released RFC7452 [9], a document designed to help engineers through the IoT design process. This document provides four basic system architectures that show the varied forms in which IoT systems are commonly found. A list of these system models is included below and diagrams are found in Appendix A [9, 20].

- Device-To-Device
- Device-To-Cloud
- Device-To-Gateway
- Back-End Data-Sharing

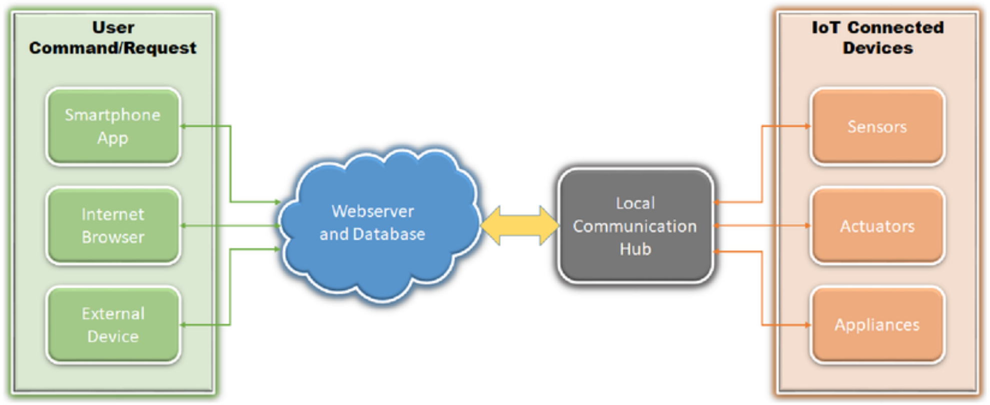


Figure 2.2: Simplified example of IoT architecture.

Another common definition of IoT architecture is the layer model which separates the components of an IoT system into functional layers [7, 21]. There are several different layer models but they all have some common sections as listed below:

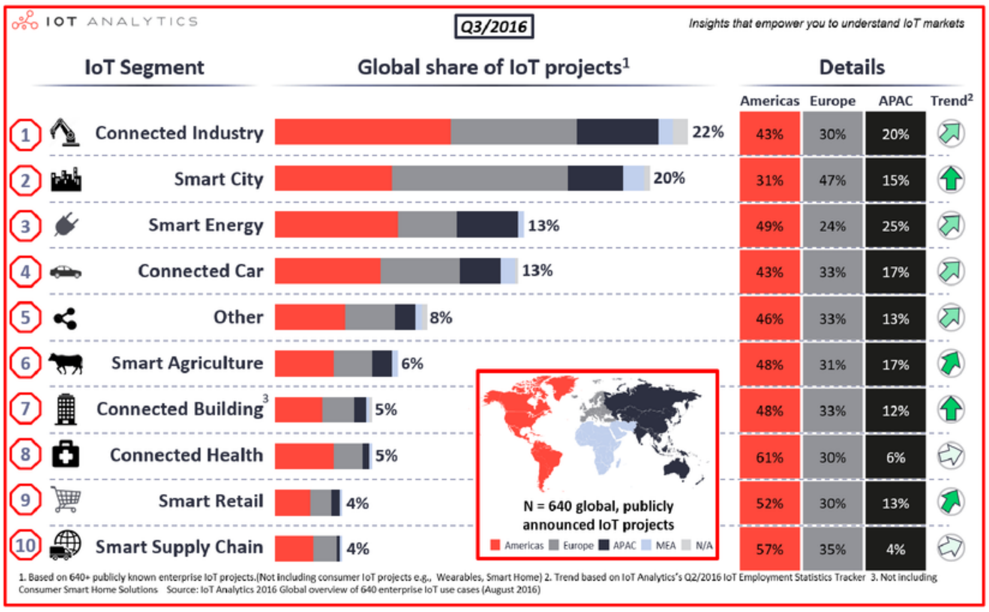
1. Application Layer - Provides user interface
2. Transportation/Network Layer - Relay data
3. Perception Layer - Gathering information, ie:sensors

IoT Market Details

The IoT market has been the focus of much interest and hype over the last few years. It is often difficult to accurately gauge the economic impact the industry will have as future market value estimates vary wildly, often by hundreds of billions of dollars. The McKinsey Global Institute, in their 2015 study, estimated that by 2025 the internet of things will add value of between \$3.9 - \$11.1 trillion USD across all major IoT impacted industries [16].

While it is almost universally agreed that the IoT will have an enormous future economic impact there is also the belief that the industry has been over-hyped. Despite this, even more conservative estimates of industry potential, such as that by BMI Research, believe that while the next five years may not yield the enormous growth forecast, the long-term potential of the IoT has been underestimated [11].

The current main areas of IoT market focus are shown in Figure 2.3. The areas shown are based on research of 640 publicly announced IoT projects and demonstrate the impact that the IoT is having across various industries. Manufacturing in particular is on the forefront of the IoT wave as focus shifts from revenue generation, through the creation of new products, to cost saving measures [10], such as connected manufacturing robots that allow for early failure detection and preventative maintenance.



	IoT Segment	Global share of IoT projects[1]	Americas	Europe	APAC	Trend[2]
1	Connected Industry	22%	43%	30%	20%	
2	Smart City	20%	31%	47%	15%	
3	Smart Energy	13%	49%	24%	25%	
4	Connected Car	13%	43%	33%	17%	
5	Other	8%	46%	33%	13%	
6	Smart Agriculture	6%	48%	31%	17%	
7	Connected Building[3]	5%	48%	33%	12%	
8	Connected Health	5%	61%	30%	6%	
9	Smart Retail	4%	52%	30%	13%	
10	Smart Supply Chain	4%	57%	35%	4%	

1. Based on 640+ publicly known enterprise IoT projects.(Not including consumer IoT projects e.g., Wearables, Smart Home) 2. Trend based on IoT Analytics's Q2/2016 IoT Employment Statistics Tracker 3. Not including Consumer Smart Home Solutions Source: IoT Analytics 2016 Global overview of 640 enterprise IoT use cases (August 2016)

Figure 2.3: IoT market areas of focus [8].

2.2 Networking and Communication Protocols

OSI Network Model

The open systems interconnection model (OSI) is a general reference model for facilitating communication between systems regardless of their individual technical characteristics. The model breaks down communications into seven layers, each of which has its own functions. It should be noted that this model is simply a reference framework and there are certain situations where components of real world applications do not fit into the designated layers. Figure 2.4 shows the seven layers and their general flow in a communication system. A complete description of the layers and their functions can be found in the official ISO documentation, ISO/IEC 7498-1 [14].

Communication Protocols

In order to facilitate reliable and secure communication between the different modules of an IoT system there are a number of tried and tested communication protocols. The use of pre-existing communication methods allows for increased interconnectivity between different IoT systems, increased security and helps avoid wasted time reinventing the wheel. The protocol that is implemented should be chosen based on the specific IoT application requirements. Some applications will prioritise low power usage while others require high data transfer rates. Commonly used protocols for the IoT include:

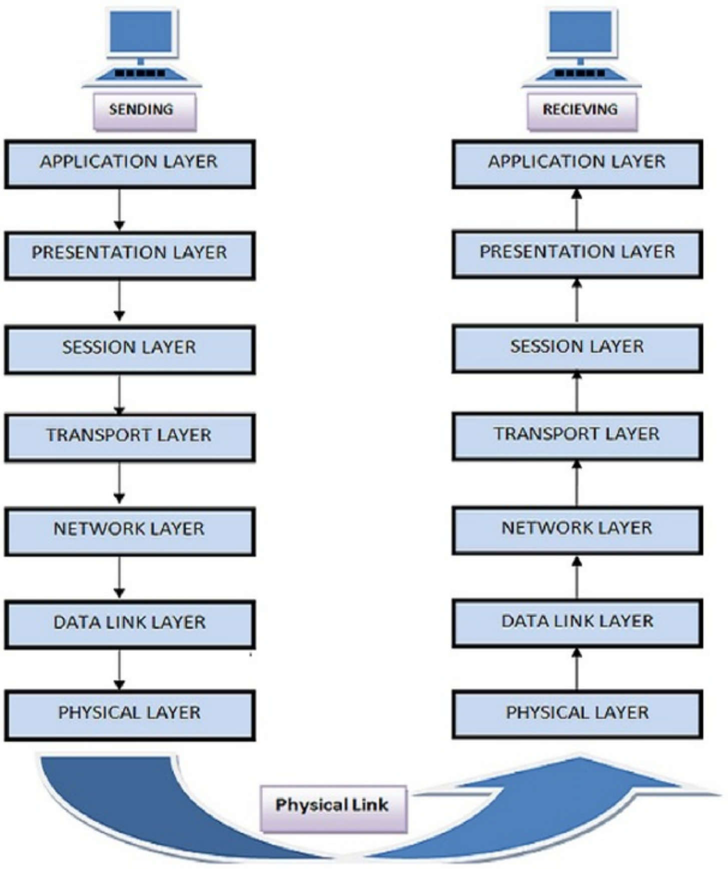


Figure 2.4: OSI Network Model [3].

- AMQP - Advanced Message Queuing Protocol
- CoAP - Constrained Application Protocol
- DDS - Data Distribution Service
- JMS - Java Message Service
- MQTT - Message Queueing Telemetry Transport
- HTTP - Hyper-Text Transfer Protocol

These communication protocols are located in the top three layers of the OSI model given in Figure 2.4 and operate on top of the transport layer which most commonly utilises the transmission control protocol (TCP) or the user datagram protocol (UDP). A good review of the pros and cons of these protocols is provided by Foster [12] and a full table of the various protocols and their characteristics is given in Appendix B.

2.3 Wireless Hardware

When creating an IoT system several considerations must be looked at when choosing the development hardware to use. Some of these considerations include cost, power requirements, physical constraints, processing power, data acquisition, communication and ease of development. The mechatronic system applied in this project will be based around the Arduino microcontroller board as it is a low cost solution with high ease of development due to the large amount of reference material available. There are four Arduino board models that include built in wireless capabilities [2]:

- Uno WiFi
- Yun
- Tian
- Industrial 101

The use of these boards allows for simple incorporation of WiFi into a mechatronic project. The trade-off of this is the increased cost associated with these more advanced boards and their relatively small number of I/O ports which may be insufficient for larger projects. In order to minimize costs for this project it was decided to combine a ESP8266 wireless module with an Arduino Mega board. This allows for low cost wireless communication with a large I/O capacity. The Arduino Mega communicates with the ESP8266 breakout board over one of it's hardware serial ports.

There are a number of potential solutions for integrating wireless connectivity into an IoT project. The selection of an appropriate device should be based on the specific requirements of the individual project. Some of the commonly available wireless solutions include:

- NRF24L01 2.4GHz Transceiver Module
- Bluetooth HC-06 Module
- Arduino WiFi Shield
- XBee ZigBee Module
- GSM/GPRS Cellular Module
- ESP8266 Module

There are also a number of other IoT hardware development platforms available that it is good to be aware of including the Raspberry Pi, Intel Edison, NodeMCU, Beaglebone Black, Particle IO, Teensy and many others. When selecting a platform it is important to consider the power, communication, cost and data acquisition requirements for an individual project in order to choose the best solution.

2.4 Cloud Services

A modern computing concept found often in studying the IoT is that of the "cloud". This term is commonly used in relation to backing up local system files such as photos on an internet server so that they can be accessed from anywhere, but what is cloud computing really? Microsoft provides the following basic explanation:

> Cloud computing is the delivery of computing services - servers, storage, databases, networking, software, analytics and more - over the internet [6]

A more formal definition is provided by the National Institute of Standards and Technology (NIST) that breaks down cloud computing into three service models, five required characteristics and four models of deployment [18].

Cloud Service Models:

- Software as a Service (**SaaS**) - User can access applications that run on the service provider's infrastructure, such as Facebook.
- Platform as a Service (**PaaS**) - User can deploy their own applications to the cloud for testing, development and use.
- Infrastructure as a Service (**IaaS**) - User can control the service provider's infrastructure as if it were their own and create a virtual data center. This allows users to avoid investment costs related to designing and purchasing their own server hardware.

Cloud Computing Characteristics:

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

Models of Deployment:

- Private cloud
- Community cloud
- Public cloud
- Hybrid cloud

In the context of this project, a web server and SQL database will be developed and deployed to a cloud server allowing for communication between the Arduino robotic system and an Android smartphone application. This usage demonstrates the platform as a service (PaaS) model of cloud computing.

2.5 Security

Cyber security is one of the most important features of connected systems. Unsecured systems can allow an attacker to potentially copy data, send false commands or take complete control of a device. In Figure 2.1 we saw that there will be a huge increase in the number of connected devices over the next decade. Without the implementation of adequate security measures many of these devices may be vulnerable to cyber-attacks.

Types of Attack

The type of attack used is often called an 'attack vector' and some of the common attack vectors that IoT designers must be aware of include [7, 21]:

- Denial of Service (DoS) - A network is deliberately sent large amounts of useless traffic in order to make the service unavailable for a user
- Malicious Code Injection - External code is 'injected' into a system to cause error or steal data (ie: SQL injection)
- Sleep Denial - Battery powered nodes are designed to sleep when not needed. This type of attack prevents a node from entering its sleep mode and results in an eventual loss of power
- Sybil attack - Multiple identities are created for a single node
- Man in the Middle - The communication channel between system components is monitored or taken control of
- Hardware tampering - Remote nodes may be physically interfered with or even replaced

There are many more possible attack vectors with new hacking techniques constantly being developed. For this reason a good understanding of cyber-security is very important for the development of IoT systems.

Security Measures

There are a number of main security measures to be aware of when considering connected devices [7, 21].

- Authentication - Devices must be authenticated before being allowed to communicate as part of a system

- Encryption - Data is encrypted to keep information confidential
- Secure Hardware - Design nodes to prevent physical tampering

A more comprehensive list of potential security breaches and their associated risks/actions is included in Appendix C [7]. This list includes an IoT layer model as referenced in Section 2.1.

Chapter 3

Project System Design Specification

3.1 Introduction and Overview

The goal of this project is the design and implementation of a mechatronic system that can be commanded over an IoT framework. It is not the goal to directly control the mechatronic system, as you would a remote control toy car, but instead simple commands are to be sent from the user to the robotic system and the system then acts autonomously based on those commands. The system also provides sensor data to be saved in a cloud-based database for access and analysis by the user. The overall project design has three main components:

1. Android application for user interaction
2. Mechatronic system
3. Cloud-based web server and data storage

The project system architecture connecting these main components is shown in Figure 3.1 along with the protocols used to communicate between each component. Note that the MySQL database is controlled using standard structured query language (SQL) commands [5] while the ESP-8266 WiFi module is controlled using an AT command set [1], derived from the Hayes command set, with the commands sent over a serial connection from the Arduino.

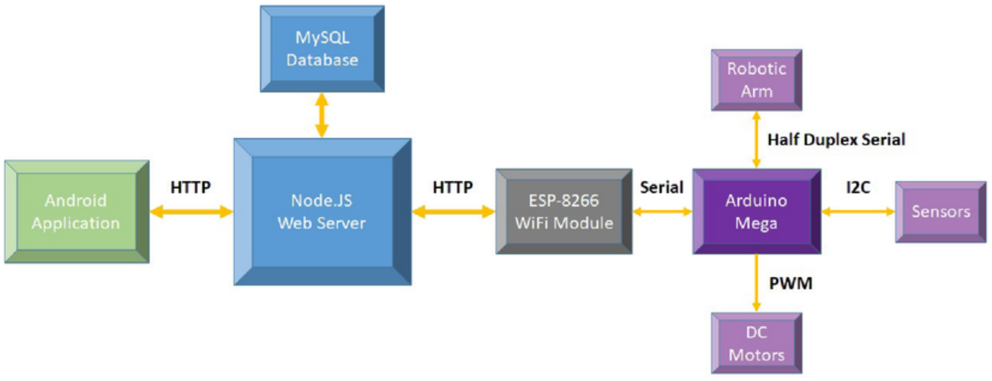


Figure 3.1: Project system architecture.

3.2 Android Application

Application Structure

The Android application acts as the user interface and controller for our system. The app was written from scratch in the Java programming language and using Android Studio software. The app includes a basic user interface for sending commands and implements swipe tabs for the various functions, for example one tab is used for inputting connection settings while another is used to send basic motor commands. Communication is achieved through the use of Android's 'HttpURLConnection' class to send asynchronous HTTP requests to the web server. It is intended that sensor data from the robotic system should be accessible to the user through the application.

3.3 Cloud Services

Web Server

The web server is implemented using the Node.js (Javascript) language and is based on the Node.js Express framework. The web server receives commands from the Android application using HTTP POST request. These commands are saved in an SQL database and the command data is then encrypted. The server waits until the next HTTP GET request is received from the Arduino system then the encrypted commands are sent to the Arduino on the GET response and an acknowledgement is sent back to the Android app confirming that the commands were received. As the server cannot initiate HTTP requests to the Android app or robotic system it must store any received commands and wait until the app or robot poll the server again.

Data Storage and Analytics

Each GET request from the Arduino will also include sensor data that is to be saved into the SQL database along with timestamp details and any other desired data. This SQL data should be accessible from the Android application to allow the user to visualise and analyse the data.

3.4 Robotic System

Design Considerations

The following important considerations were taken into account during the design process for the robotic system:

- System must be able to receive control commands from an Android application over the internet and to acknowledge receipt of commands.
- System must be able to send sensor data to web server for data analysis purposes.
- Chassis design should be modular to allow for easy assembly/disassembly and future redesign as well as ease of 3D printing.
- System should be mechanically stable when robotic arm is mounted to chassis.
- System should be battery powered.
- System should be mobile and implement autonomous collision avoidance behaviours.
- Cost should be minimized.

System Components

The chosen components for the mechatronic system are:

- 3D printed chassis
- 4 x 6V DC motors
- 2 x 12V DC motor driver board
- Arduino Mega 2560
- ESP-8266 WiFi module
- Crustcrawler AX-12 robotic arm
- Ultrasonic sensors
- 12V 2.2 AH SLA battery
- Step down DC-DC converter

Peripheral Control and Communication

The Arduino Mega controls and communicates with all the peripheral components based on various protocols as described below:

- ESP8266 - AT commands are sent from the Arduino over one of the hardware serial ports. The system polls the web server at regular intervals to see if any updated commands have been received from the Android app user.
- Robotic Arm - The Crustcrawler AX-12 robotic arm utilises half-duplex serial communication as opposed to the full-duplex used by the Arduino. In order to translate between full and half duplex, extra circuitry is required in the form of a 74LS241 tri-state buffer. Stable communication was achieved at a baud rate of 200,000 bps.
- Sensors - Ultrasonic sensors readings are taken using the I2C protocol.
- DC Motors - The motors are controlled using pulse width modulation (PWM) by using the analogWrite() command on Arduino PWM pins connected to the motor driver breakout boards.

Chapter 4

Completed System Specification

This chapter details the completed IoT system that was eventually created for this project. It should be noted that there are a number of differences between the intended design functionality detailed in Chapter 3 and the final system. These changes are described in detail below.

4.1 Android Application

The Android application was written in the java programming language using Android Studio software. This application functions as the user interface for the overall IoT framework. User commands are input through the use of push-buttons, slide-bars and text prompts. Prior to undertaking this project I had minimal experience in Android app programming and so this was a difficult and time consuming learning experience. Due to my lack of experience the application produced is functional but does not follow Android's recommended best programming practices and cannot be considered complete or fully secure in its current state.

In its basic form the application converts the user inputs into a string and using the 'HttpURLConnection' class provided by Android to asynchronously transmit the information to the web server using HTTP POST functions. Due to time constraints I was unable to add functionality to the app to allow recorded robot sensor data to be viewed by the user. Screen-shots of the final application user interface are included below.

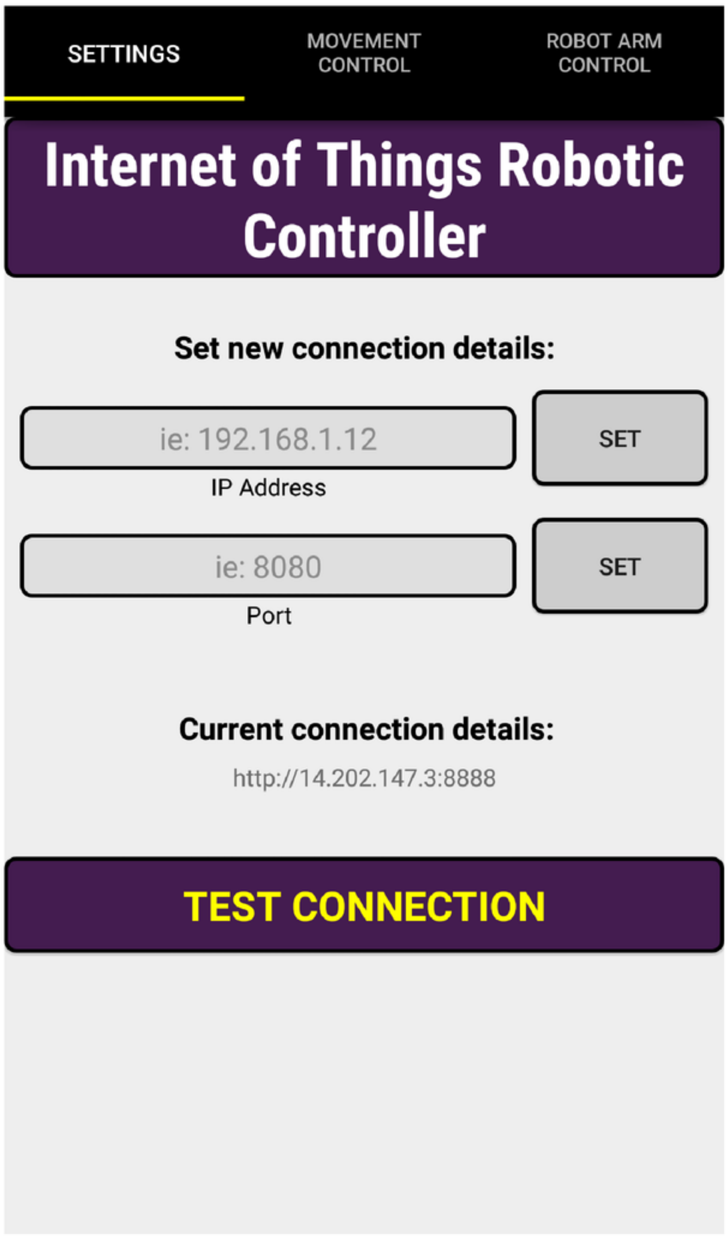


Figure 4.1: Tab 1 - Connection configuration

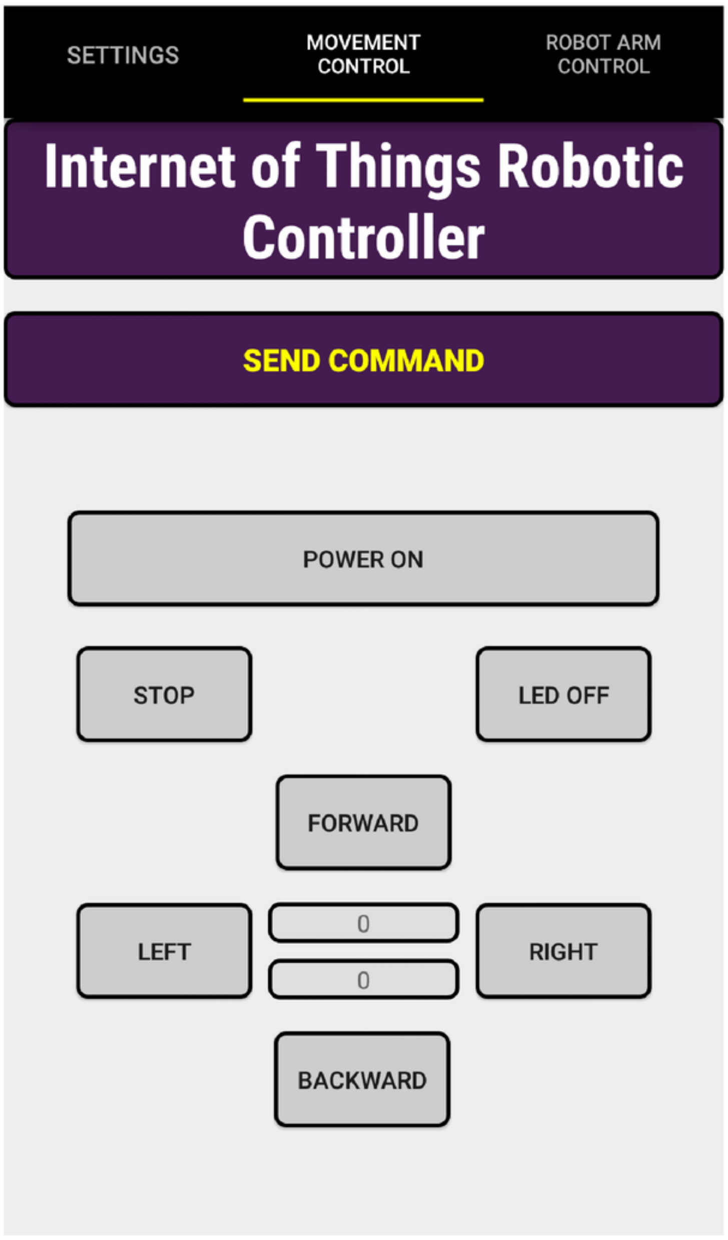


Figure 4.2: Tab 2 - Movement control

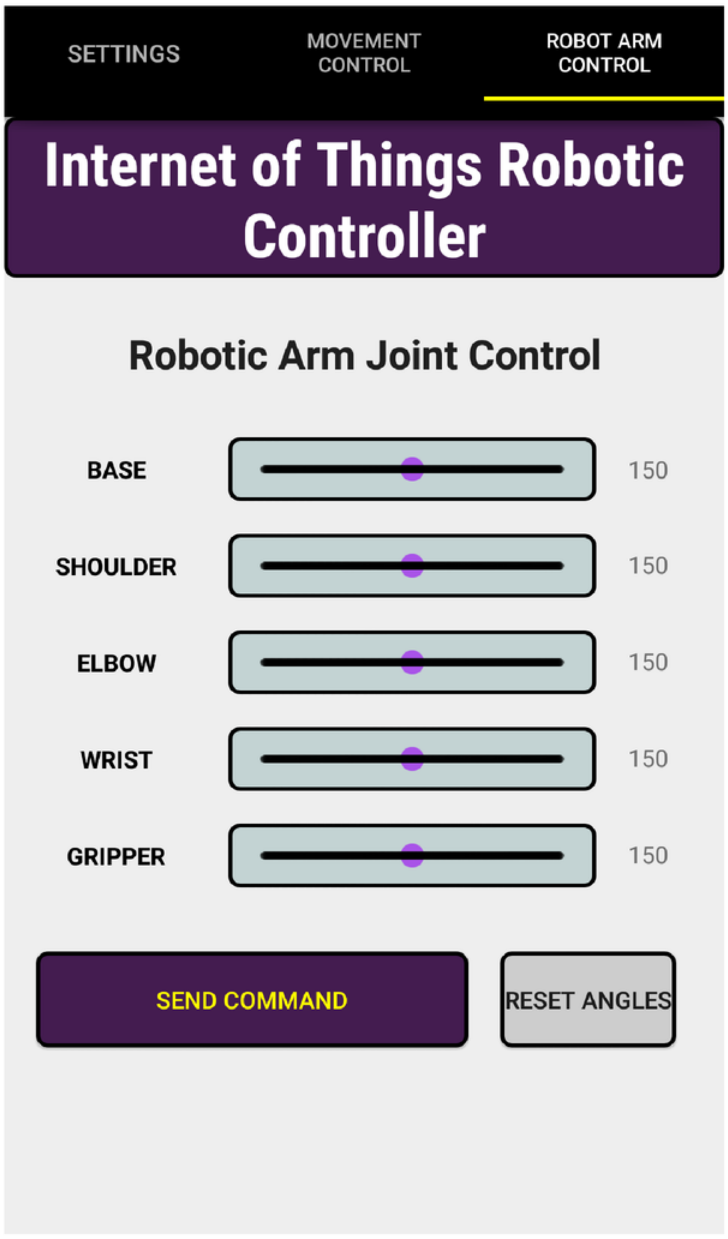


Figure 4.3: Tab 3 - Robotic arm control

4.2 Cloud Services

The cloud component of this project involved the development of a web server and database that would facilitate communication between the user interface application and the robot as well as recording data for later analysis. Originally it had been intended to deploy the web server to an existing cloud server such as OpenShift, Microsoft Azure or Google Cloud, but I ran into networking issues when trying to connect to these servers using the ESP8266 WiFi module. The server was instead hosted on a Beaglebone Black development board running a Debian operating system. The server was connected to my home network and was accessible after configuring port forwarding of the relevant port on my router.

The asynchronous language Node.js, a derivative of javascript, was used for this section of the project. One of the major challenges faced during this stage of the project was learning how to program in an asynchronous language. Asynchronous code is also called 'non-blocking' code where functions can all execute at the same time instead of one after the other. This requires a much different approach than conventional languages where code is executed line by line.

The second major challenge faced in this section was in finding an encryption and authorisation algorithm that could be implemented by both the web server and the robot. After significant testing, development and debugging I was able to implement the Data Encryption Standard (DES) algorithm. Unfortunately this is an outdated algorithm that provides only basic security. Ideally I would implement ChaCha20 encryption with Poly1305 authentication, but in attempting to do this i ran into significant problems and elected to keep the basic encryption to stay on schedule. The complexity of encryption algorithms made the process difficult but again it was a valuable learning experience. A flow chart of the functional process of the web server is shown by Figure 4.4 and the code used to encrypt my data is included in Figure 4.5.

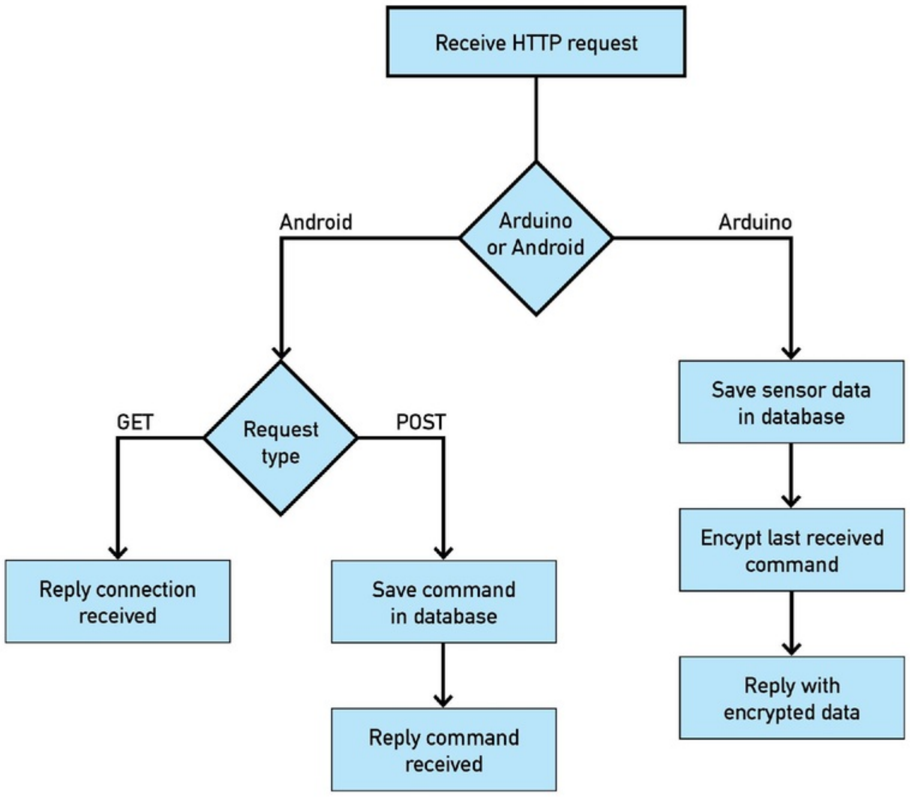
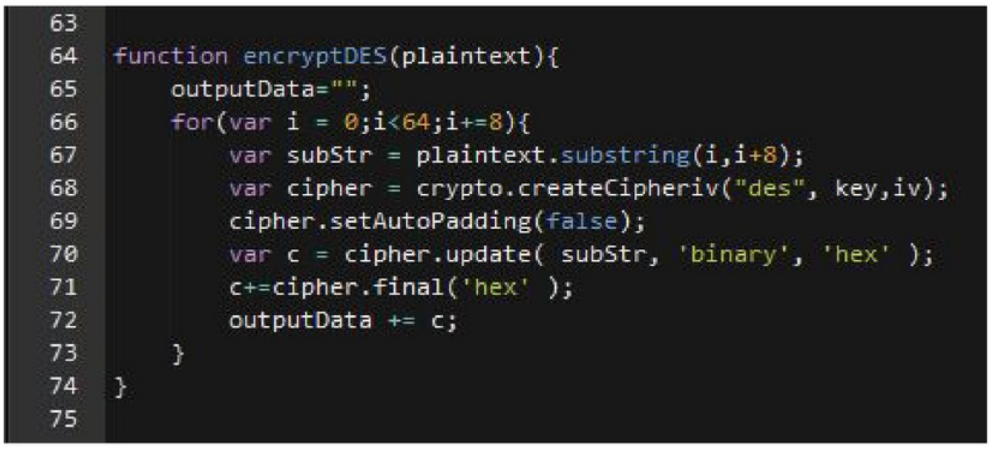


Figure 4.4: Web server process flow chart



```
function encryptDES(plaintext){
    outputData="";
    for(var i = 0;i<64;i+=8){
        var subStr = plaintext.substring(i,i+8);
        var cipher = crypto.createCipheriv("des", key,iv);
        cipher.setAutoPadding(false);
        var c = cipher.update( subStr, 'binary', 'hex' );
        c+=cipher.final('hex' );
        outputData += c;
    }
}
```

Figure 4.5: Web server DES encryption function

4.3 Robotic System

The mechatronic system developed took the form of a four wheeled, battery powered robot with a Crustcrawler AX-12 robotic arm mounted to it. The intention had been to implement autonomous behaviours in the robot but due to time constraints this was not possible and focus was instead placed on developing the IoT framework.

Chassis

The robot chassis was designed from scratch using Autocad software and as I had little CAD experience it is a very simple design. As the chassis was to be 3d printed it was required that all chassis components be small enough to fit on the print bed of my available 3d printer (20cm x 20cm). Due to this limitation the chassis design is in several parts which join together as shown below. The chassis components were printed using PLA plastic. The design was intended to keep the centre of gravity low and use a heavy sealed lead acid (SLA) battery to help keep the robot stabilised when the robotic arm is attached. Photos of the completed robotic system can be seen in the appendices.

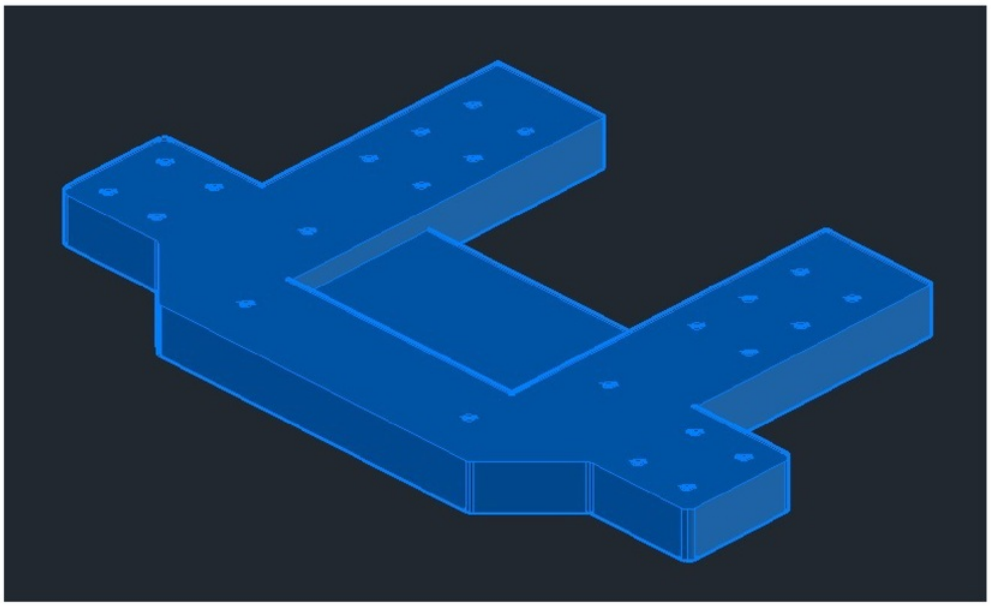


Figure 4.6: Base component

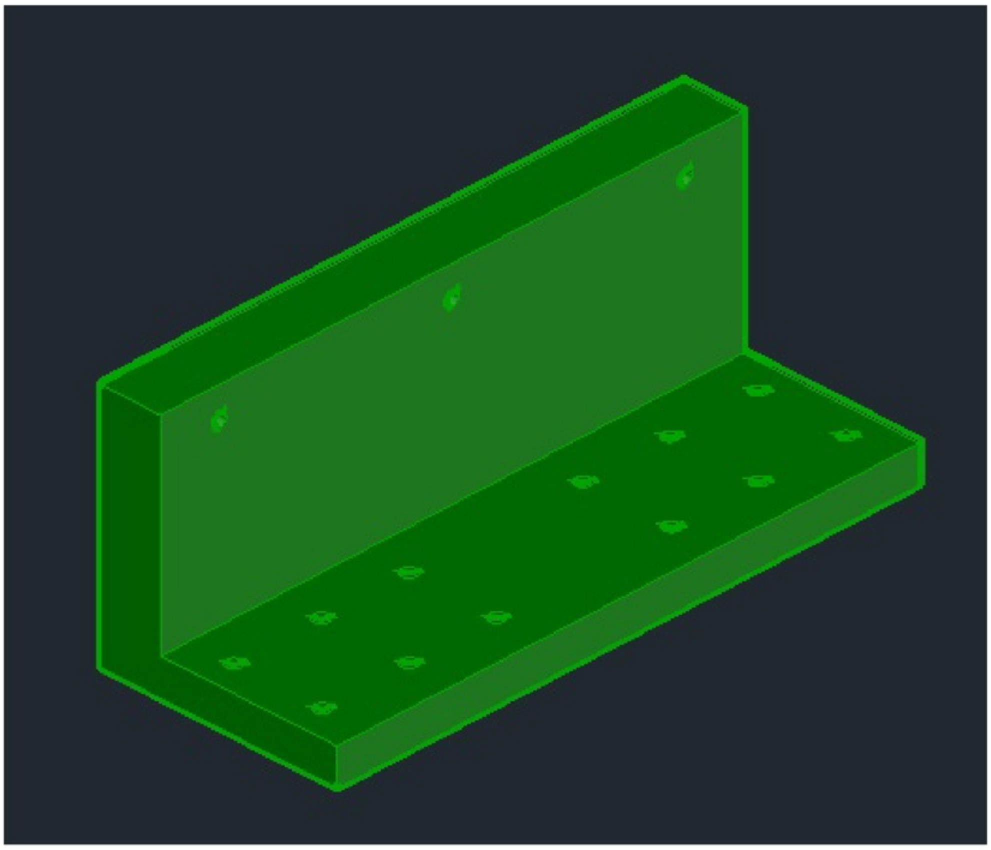


Figure 4.7: Side attachment

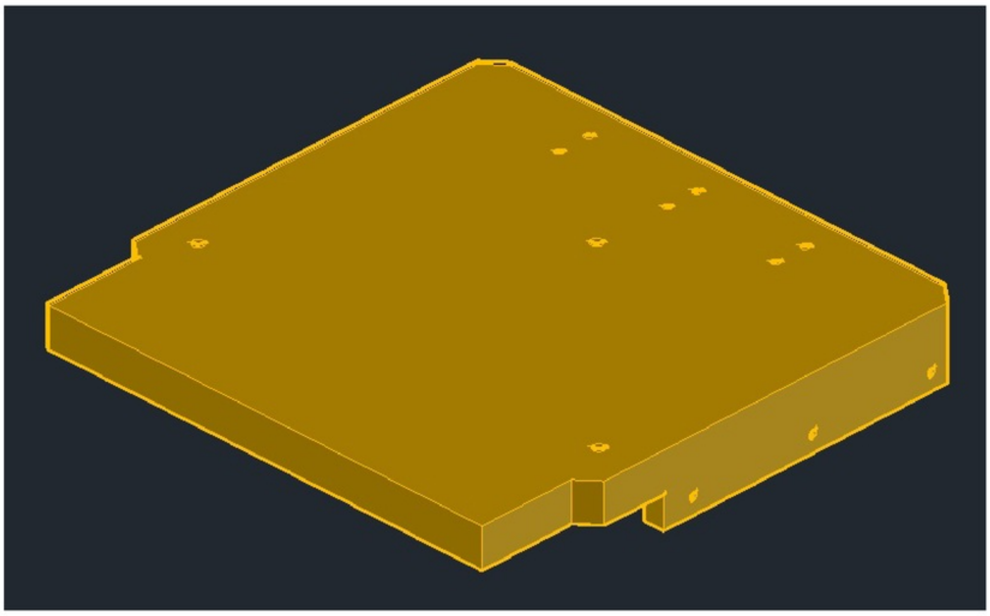


Figure 4.8: Robotic arm mount

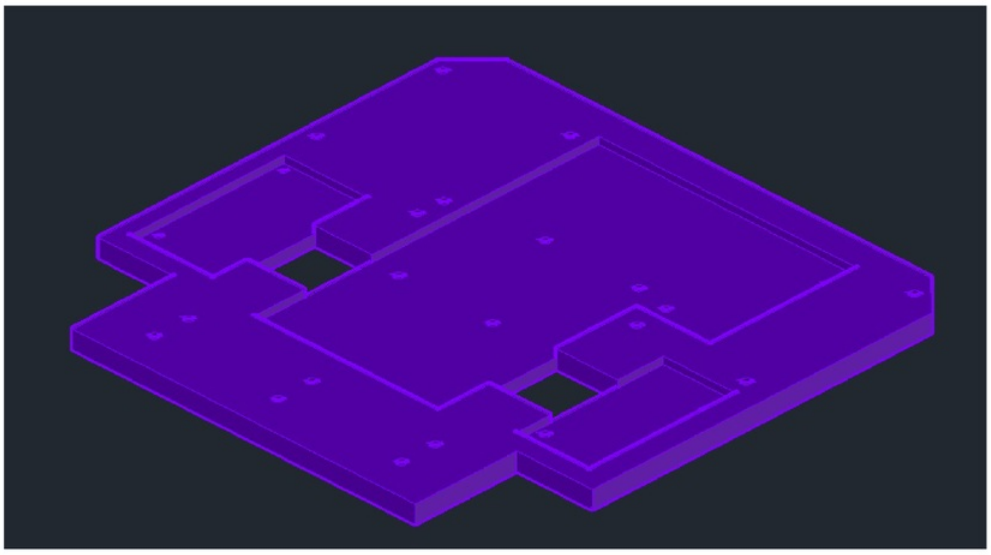


Figure 4.9: Electrical components mount

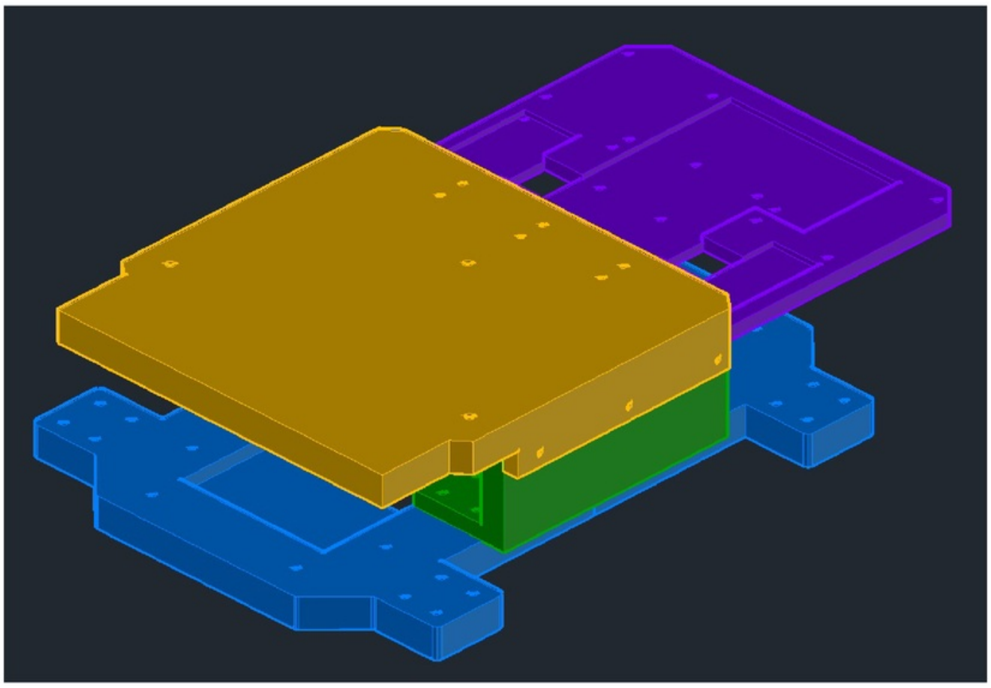


Figure 4.10: Complete model of designed components

Electronics

The core of the electronic system is the Arduino Mega board. This board controls the motors and robotic arm as well as reading sensor data and communicating with the ESP8266 Wifi module. The full list of electronic components used includes:

- Arduino Mega
- ESP8266 WiFi module
- 2 x DC-DC voltage converter
- 4 x 6V DC motor with encoder
- 12V 2.2AH SLA battery
- 2 x Motor driver board
- 74LS241 Tri-state buffer IC

Due to time constraints I was unable to implement ultrasonic sensors in the final product as intended. Instead the motor encoder values were used as sensor values when testing the system. The two DC-DC voltage converters take the 12V battery as an input and output 5V and 9V. The 5V supply is used to power the ESP8266 module as it requires higher current than can be supplied by the Arduino 5V rail. The 9V supply is used to power the Arduino via the barrel jack connection. The 12V battery is used to directly power the motors as well as robotic arm servos. A circuit diagram of the system is provided in Figure 4.11. Note that this diagram does not include the motor encoder connections or the circuit for controlling the robotic arm. Instructions for interfacing Dynamixel servos with an Arduino can be found online [19].

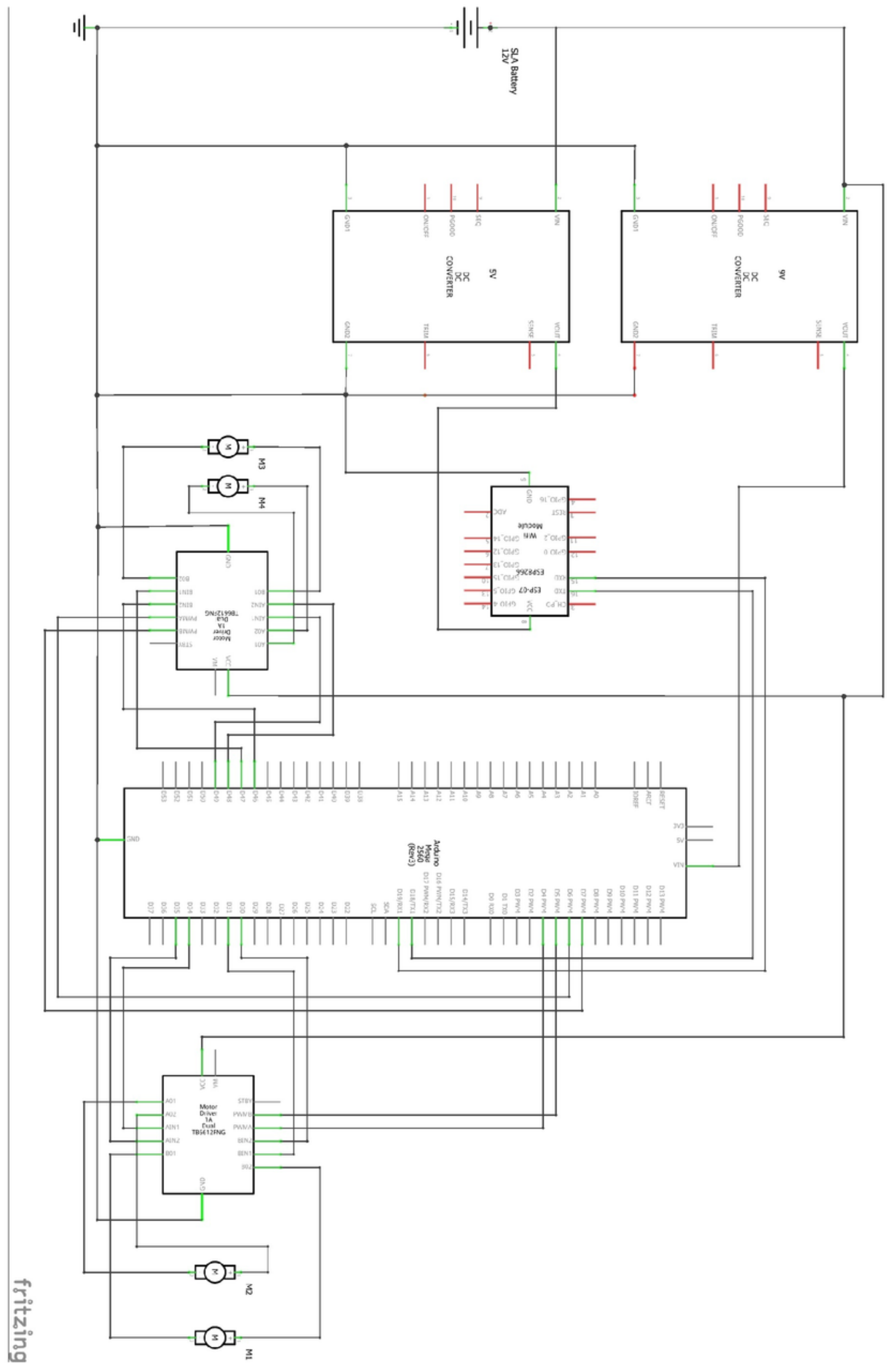


Figure 4.11: Electrical circuit diagram

Chapter 5

System Experimental Analysis

This chapter details the experimental analysis performed on the completed IoT system. Three main areas were examined during this process to test the stability and efficiency of the system as a whole. These areas were:

- Data Analytics - Testing if commands and sensor data were accurately recorded into SQL database for later analysis
- Security Algorithms - Testing the performance of various data encryption/decryption algorithms on the Arduino microcontroller
- Spatial Accuracy - Testing the spatial accuracy of the robotic system when given a move command

Data Analytics

One of the major driving aspects behind the IoT is the desire for large amounts of data for analysis purposes. IoT systems must be able to record data so that usage patterns or other metrics may be analysed. In this project an SQL database was used to capture sensor data from the robot as well as command data sent from the user. Both sets of data are stored in a database table.

For this experiment I used an encoder attached to the robot motor as the example sensor, while the IP address, time-stamp and command values were saved from any Android app user commands that were received. The 'CmdReceived' column is the Arduino's confirmation to the server that it received the last command. In order to test the robustness of the data storage system several test runs were carried out.

The experimental data from two of these test runs is included below. Table 5.1 shows the results of a test where the recorded user commands and sensor data were both successfully recorded. Table 5.2 displays an incomplete table, indicating a failure of some kind.

After analysing the performance of my system I concluded that a flaw in my design was to blame. As part of the design the Arduino polls the web server at most every 2.5 seconds. Due to this fact, if commands are received at a similar or faster rate than this

Table 5.1: MYSQL database successful

ID	Address	Timestamp	Speed	Angle	Power	Encoder1	CmdReceived
1	::ffff:49.195.94.91	2017-11-04T16:38:38	0	0	0	122	1
2	::ffff:49.195.94.91	2017-11-04T16:38:46	0	0	0	122	1
3	::ffff:49.195.94.91	2017-11-04T16:38:55	0	0	0	143	1
4	::ffff:49.195.94.91	2017-11-04T16:39:04	0	0	0	212	1
5	::ffff:49.195.94.91	2017-11-04T16:39:11	0	0	0	245	1
6	::ffff:49.195.94.91	2017-11-04T16:39:20	0	0	0	222	1
7	::ffff:49.195.94.91	2017-11-04T16:39:26	0	0	0	196	1
8	::ffff:49.195.94.91	2017-11-04T16:39:34	0	0	0	171	1
9	::ffff:49.195.94.91	2017-11-04T16:39:44	0	0	0	171	1
10	::ffff:49.195.94.91	2017-11-04T16:39:51	0	0	0	171	1

Table 5.2: MYSQL database failure

ID	Address	Timestamp	Speed	Angle	Power	Encoder1	CmdReceived
1	::ffff:49.195.94.91	2017-11-04T16:45:59	0	0	0	20	1
2	::ffff:49.195.94.91	2017-11-04T16:46:08	0	0	0	46	1
3	::ffff:49.195.94.91	2017-11-04T16:46:18	0	0	0	68	1
4	::ffff:49.195.94.91	2017-11-04T16:46:19	0	0	0	99	1
5	::ffff:49.195.94.91	2017-11-04T16:46:20	0	0	0	132	1
6	::ffff:49.195.94.91	2017-11-04T16:46:20	0	0	0	-	-
7	::ffff:49.195.94.91	2017-11-04T16:46:21	0	0	0	-	-
8	::ffff:49.195.94.91	2017-11-04T16:46:22	0	0	0	-	-
9	::ffff:49.195.94.91	2017-11-04T16:46:23	0	0	0	-	-
10	::ffff:49.195.94.91	2017-11-04T16:46:24	0	0	0	-	-

polling interval then multiple sets of command data may be written into the database but the Arduino only responds to the last command that was received before polling the server again. This means that some commands are incorrectly marked as received when they were actually missed. In order to correct this system flaw some further software development and debugging is required. One quick potential fix is to block the Android application from sending new commands for several seconds after the last command was sent. This solution would just be a patch though and not address the underlying issues.

Table 5.3: Cryptographic algorithm experimental results

Algorithm	Encrypt/Byte (uS)	Decrypt/Byte (uS)	Key Set (uS)
AES-128-ECB	33.75	67.84	365.43
AES-192-ECB	40.55	82.18	405.05
AES-256-ECB	47.34	96.53	412.66
ChaCha8 128-bit	9.66	9.66	42.99
ChaCha8 256-bit	9.66	9.66	41.91
ChaCha12 128-bit	12.33	12.33	42.99
ChaCha12 256-bit	12.33	12.33	41.91
ChaCha20 128-bit	17.67	17.68	42.98
ChaCha20 256-bit	17.67	17.68	41.91
ChaChaPoly1305	43.59	43.58	978.67
Speck-128-ECB	9.75	10.12	253.99
Speck-192-ECB	10.03	10.42	264.68
Speck-256-ECB	10.32	10.71	275.31
DES	79.22	52.88	2101

Security Algorithms

When considering the security algorithms to use there were three major considerations taken into account that are listed below in the order of importance:

1. Ease of implementation
2. Performance on Arduino microcontroller
3. Level of security

Initially I had chosen performance as the most important factor but due to my lack of experience with cyber-security algorithms prior to this project I was forced to change my priorities. During the testing phase of the project I tested a number of different algorithms using the ArduinoLibs cryptographic library [22]. The Arduino mega was used as the test device and the results obtained are detailed in Table 5.3.

It can be seen from these results that the chosen DES algorithm performs poorly in comparison to its modern counterparts. This is because the DES algorithm was first created in the 1970's. DES is considered unsafe for new applications but was the easiest algorithm for me to implement in practice on both the Arduino and Node.js platforms. Furthermore the DES algorithm does not implement any authentication such as Poly1305, and so it only provides a basic level of security to the system. I did not feel it was necessary to record encryption algorithm data for the web server as it has access to much greater processing power than the Arduino and so encryption times would be negligible for the small amounts of data being transmitted.

Table 5.4: Movement command experimental data

Set Distance (mm)	Measured Distance (mm)	Set Angle (deg)	Measured Angle (deg)
100	94	45	41
200	179	90	83
500	428	180	173
1000	832	360	337
-100	-97	-45	-48
-200	-183	-90	-84
-500	-444	-180	-169
-1000	-898	-360	-337

Spatial Accuracy

The final area of testing was that of the accuracy of the robot at executing any given move commands. The two types of movement commands recognised are a straight forward/back move of user defined length and a left/right turn of some multiple of 45 degrees. As the major focus of this project was on the development of the IoT communication framework, the robot control system has only been designed to provide a basic level of accuracy.

Using the encoder rating of 341 pulses per revolution and a measured wheel diameter of 65mm gives us approximately 20.42cm per revolution, or 0.6mm per encoder pulse. Using these figures along with some trial and error allowed me to achieve the results shown in Table 5.4 when testing movement commands. Note that the measurements shown were made using a tape measure and protractor and each measurement was averaged over 3 runs at each distance/angle. Measurements were also taken when the robotic arm was not mounted to the chassis.

The results show the poor spatial accuracy of the robot as expected. Reasons for this inaccuracy include a simple control system, poor quality motors and encoders, slippage and lack of angular sensor feedback. In order to improve the quality and spatial accuracy of this system a number of tasks should be undertaken.

- Implement PID controllers
- Use higher quality encoders
- Addition of inertial measurement unit to provide angular feedback
- Further develop code to prevent motor shutoff function from being blocked by other running functions (such as receiving and processing new command data)

Chapter 6

Conclusions and Future Work

6.1 Reflections

This thesis has been the culmination of my four years of study in the field of mechatronics. While I did not meet all the original goals I set out to achieve for this project I am content as I feel I have learned an enormous amount in the process. I have developed new skills in CAD modelling, mobile application development, networking and security theory, 3D printing and asynchronous programming which will serve me well in my future career. I am proud of the work I have completed and of the system that I created.

6.2 Future work

As it stands the IoT framework I have created is functional but by no means is it a completed system. The following tasks describe what my next steps would be were I to continue with the system development.

- Modify Android app to meet Android's recommended best programming practices
- Re-design and reprint the chassis for increased strength, stability and improved aesthetics
- Run the web server from an existing cloud platform rather than through my personal hardware and home network
- Implement more complete security for the system including an up to date encryption and authentication algorithm.
- Add sensors such as inertial motion units to the robot chassis to aid with autonomous behaviour
- Improve Arduino code to implement more robust communication and improve stability

- Add ability for user to view and analyse data recorded in SQL database

These are just some of the possible steps that could be taken to continue the development of this IoT controlled mechatronic system.

6.3 Conclusion

During this project I have developed from scratch a new and functional IoT controlled mechatronic system. The system brings together a user friendly Android application, a web server for communication and data storage, and a robotic platform for command execution and data gathering. During the creation of these components I gained a wide range of new skills and knowledge that will be invaluable in the future. I was unable to meet some of the original design requirements due to underestimating the time required to learn so many new skills, but despite this I am more than satisfied with the system I created.

The final system has several shortcomings including inadequate security measures and an imperfect data collection method, but with some minor work many of these problems could be quickly rectified. In undertaking any future projects of this scope I will have a much better appreciation of the need for strict time management and the value of spending extra effort during the research and design phase.

To summarise, the outcome of this project was the successful production of a new and functional Internet of Things controlled mechatronic system. While several of the original design criteria were not met due unforeseen technical problems and time management issues, with a little work the system can be readily modified into a more complete, robust and secure system.

Chapter 7

Abbreviations

IoT	Internet of Things
IAB	Internet Architecture Board
HTTP	Hyper-Text Transfer Protocol
HTML	Hyper-Text Markup Language
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
IP	Internet Protocol
WWW	World Wide Web
M2M	Machine to Machine
ISO	International Organisation for Standardization
OSI	Open Systems Interconnection
API	Application Programming Interface
NIST	National Institute of Standards and Technology
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
SQL	Structured Query Language
PWM	Pulse Width Modulation
DES	Data Encryption Standard
SLA	Sealed Lead Acid

Appendix A

IoT System Models

A.1 Device-To-Device

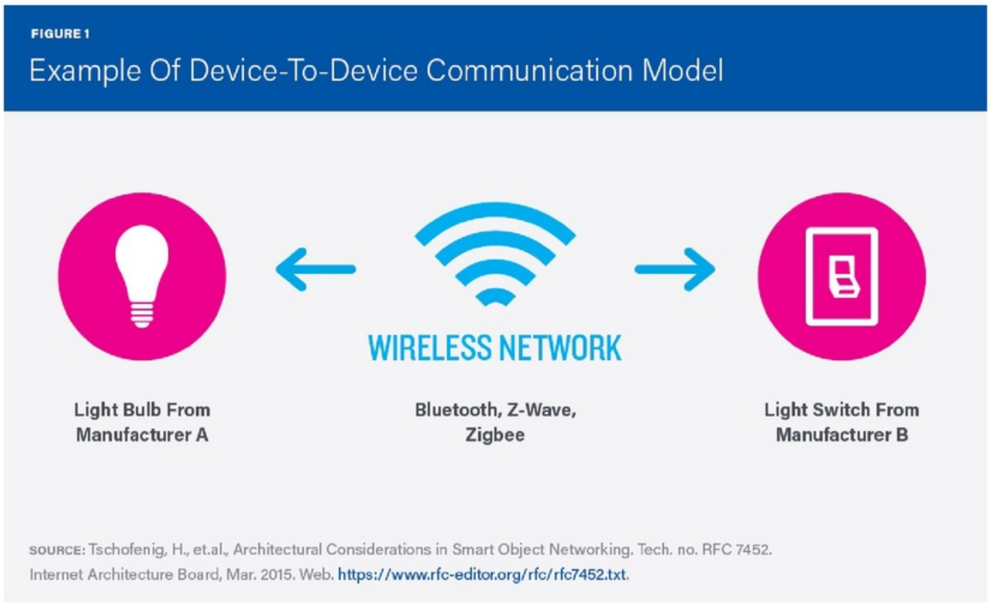


FIGURE 1
Example Of Device-To-Device Communication Model
WIRELESS NETWORK
Light Bulb From Manufacturer A
Bluetooth, Z-Wave, Zigbee
Light Switch From Manufacturer B
SOURCE: Tschofenig, H., et.al., Architectural Considerations in Smart Object Networking. Tech. no. RFC 7452. Internet Architecture Board, Mar. 2015. Web. https://www.rfc-editor.org/rfc/rfc7452.txt.

Figure A.1: Device-To-Device system model [20]

.

A.2 Device-To-Cloud

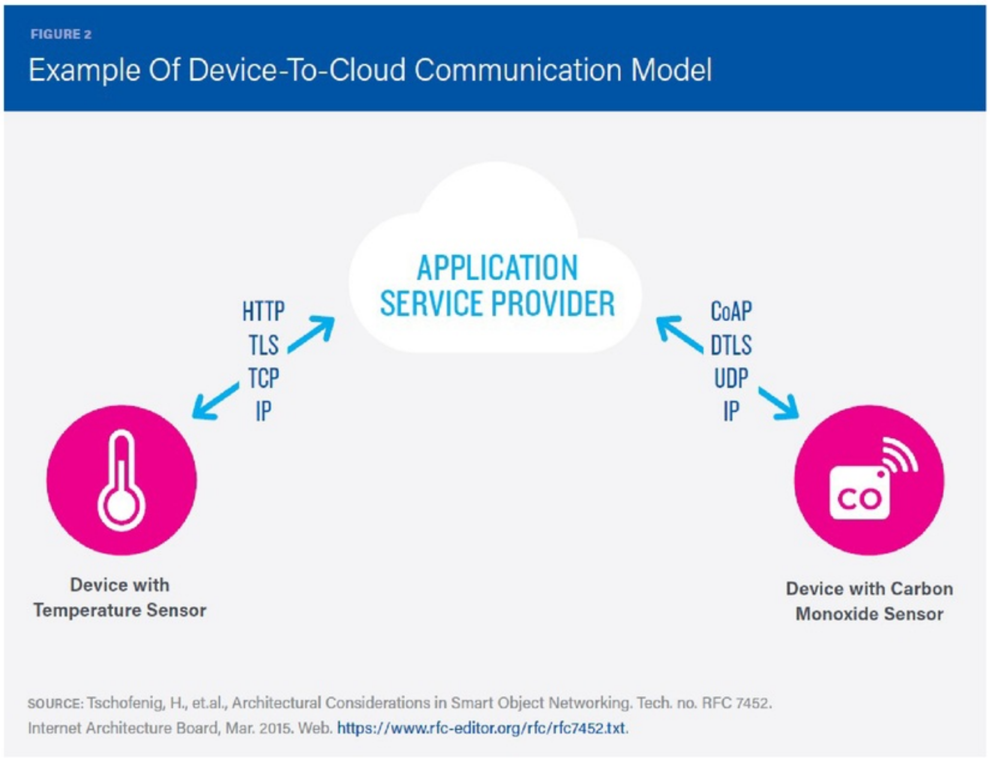


FIGURE 2
Example Of Device-To-Cloud Communication Model
APPLICATION SERVICE PROVIDER
HTTP
TLS
TCP
IP
CoAP
DTLS
UDP
IP
CO
Device with Temperature Sensor
Device with Carbon Monoxide Sensor
SOURCE: Tschofenig, H., et.al., Architectural Considerations in Smart Object Networking. Tech. no. RFC 7452. Internet Architecture Board, Mar. 2015. Web. https://www.rfc-editor.org/rfc/rfc7452.txt.

Figure A.2: Device-To-Cloud system model [20]

.

A.3 Device-To-Gateway

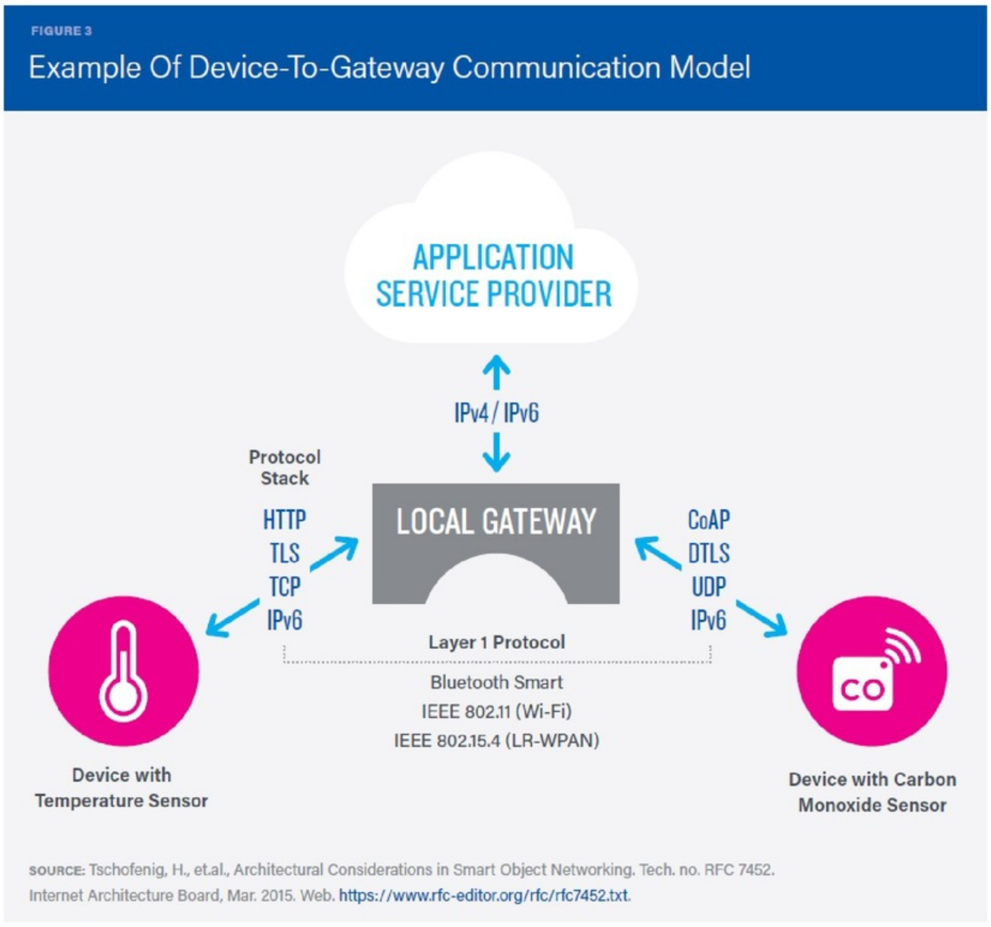


FIGURE 3
Example Of Device-To-Gateway Communication Model
APPLICATION SERVICE PROVIDER
IPv4 / IPv6
Protocol Stack
HTTP
TLS
TCP
IPv6
LOCAL GATEWAY
CoAP
DTLS
UDP
IPv6
Layer 1 Protocol
Bluetooth Smart
IEEE 802.11 (Wi-Fi)
IEEE 802.15.4 (LR-WPAN)
CO
Device with Temperature Sensor
Device with Carbon Monoxide Sensor
SOURCE: Tschofenig, H., et.al., Architectural Considerations in Smart Object Networking. Tech. no. RFC 7452. Internet Architecture Board, Mar. 2015. Web. https://www.rfc-editor.org/rfc/rfc7452.txt.

Figure A.3: Device-To-Gateway system model [20]

.

A.4 Back-End Data-Sharing

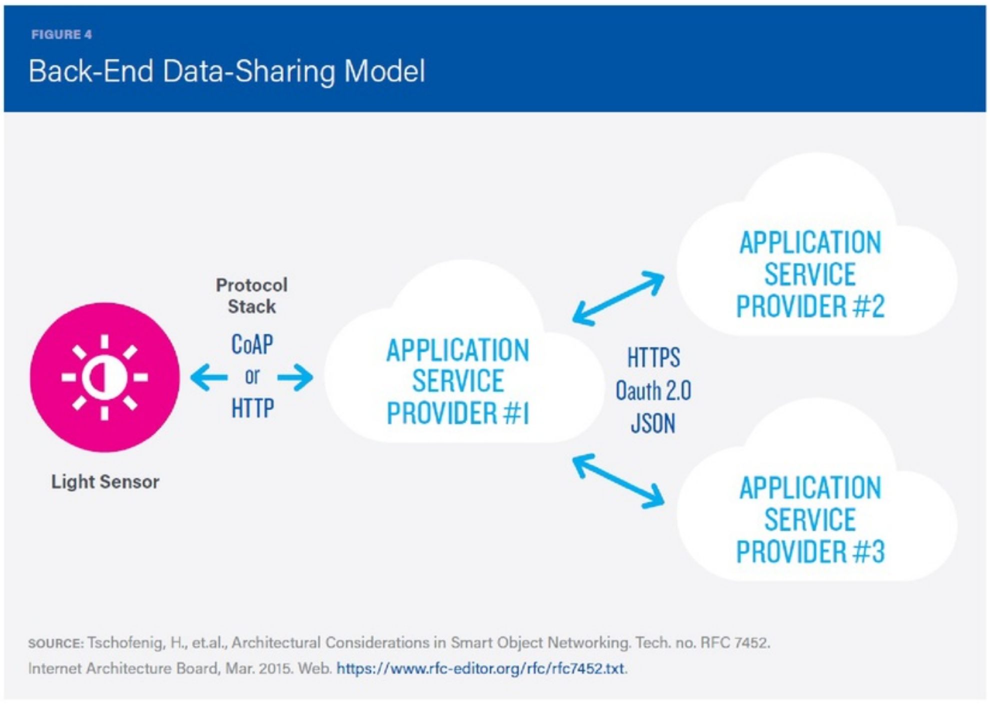


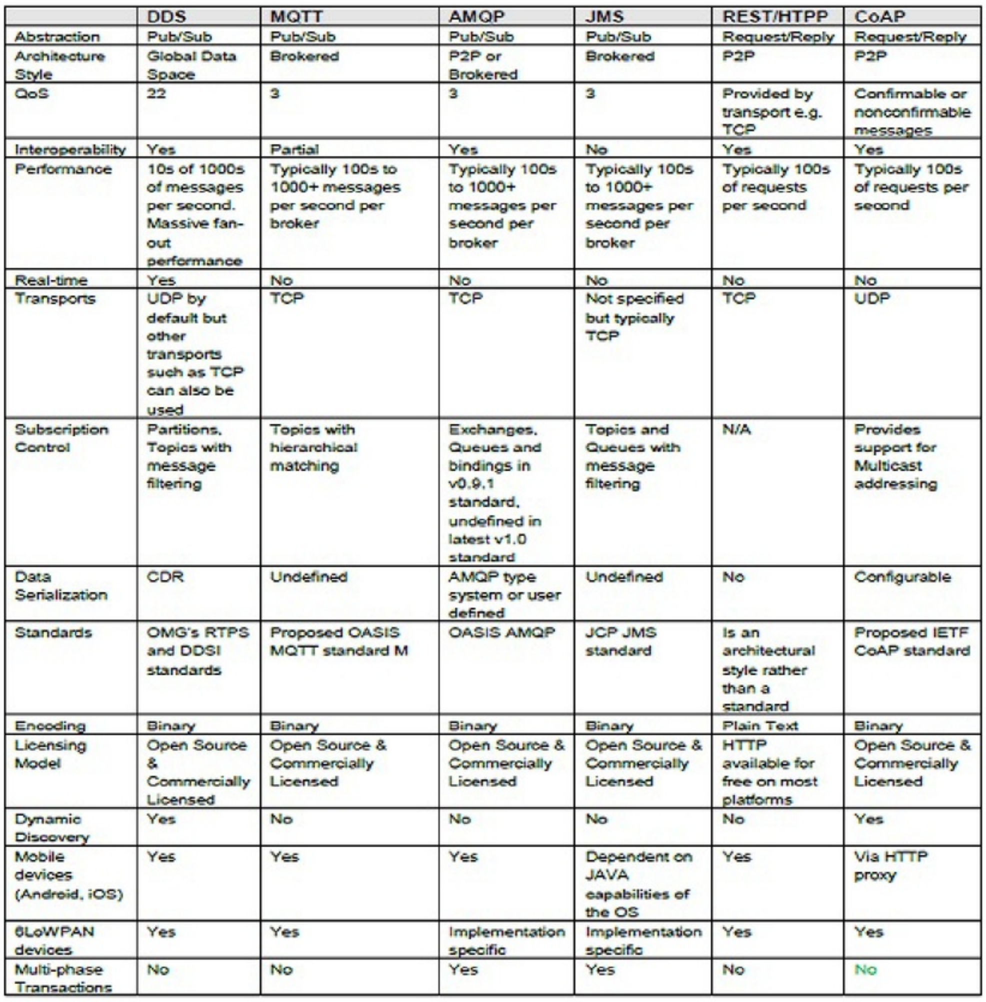
FIGURE 4
Back-End Data-Sharing Model
Light Sensor
Protocol Stack
CoAP or HTTP
APPLICATION SERVICE PROVIDER #1
HTTPS
Oauth 2.0
JSON
APPLICATION SERVICE PROVIDER #2
APPLICATION SERVICE PROVIDER #3
SOURCE: Tschofenig, H., et.al., Architectural Considerations in Smart Object Networking. Tech. no. RFC 7452. Internet Architecture Board, Mar. 2015. Web. https://www.rfc-editor.org/rfc/rfc7452.txt.

Figure A.4: Back-End Data-Sharing system model [20]

.

Appendix B

IoT Communication Protocols



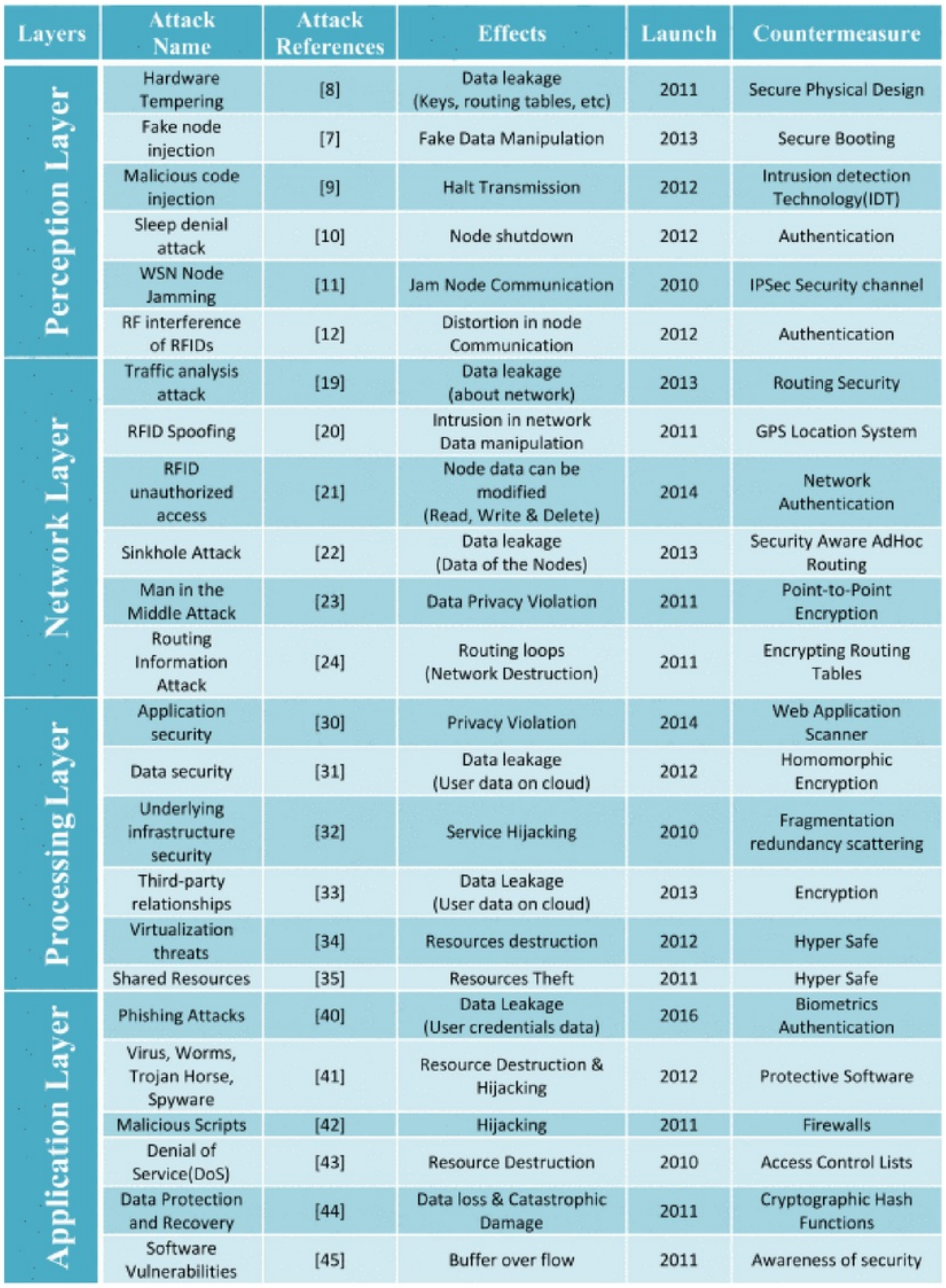
	DDS	MQTT	AMQP	JMS	REST/HTPP	CoAP
Abstraction	Pub/Sub	Pub/Sub	Pub/Sub	Pub/Sub	Request/Reply	Request/Reply
Architecture Style	Global Data Space	Brokered	P2P or Brokered	Brokered	P2P	P2P
QoS	22	3	3	3	Provided by transport e.g. TCP	Confirmable or nonconfirmable messages
Interoperability	Yes	Partial	Yes	No	Yes	Yes
Performance	10s of 1000s of messages per second. Massive fan-out performance	Typically 100s to 1000+ messages per second per broker	Typically 100s to 1000+ messages per second per broker	Typically 100s to 1000+ messages per second per broker	Typically 100s of requests per second	Typically 100s of requests per second
Real-time	Yes	No	No	No	No	No
Transports	UDP by default but other transports such as TCP can also be used	TCP	TCP	Not specified but typically TCP	TCP	UDP
Subscription Control	Partitions, Topics with message filtering	Topics with hierarchical matching	Exchanges, Queues and bindings in v0.9.1 standard, undefined in latest v1.0 standard	Topics and Queues with message filtering	N/A	Provides support for Multicast addressing
Data Serialization	CDR	Undefined	AMQP type system or user defined	Undefined	No	Configurable
Standards	OMG's RTPS and DDSI standards	Proposed OASIS MQTT standard M	OASIS AMQP	JCP JMS standard	Is an architectural style rather than a standard	Proposed IETF CoAP standard
Encoding	Binary	Binary	Binary	Binary	Plain Text	Binary
Licensing Model	Open Source & Commercially Licensed	Open Source & Commercially Licensed	Open Source & Commercially Licensed	Open Source & Commercially Licensed	HTTP available for free on most platforms	Open Source & Commercially Licensed
Dynamic Discovery	Yes	No	No	No	No	Yes
Mobile devices (Android, iOS)	Yes	Yes	Yes	Dependent on JAVA capabilities of the OS	Yes	Via HTTP proxy
6LoWPAN devices	Yes	Yes	Implementation specific	Implementation specific	Yes	Yes
Multi-phase Transactions	No	No	Yes	Yes	No	No

Figure B.1: Summary of IoT communication protocols [12]

.

Appendix C

Security Measures



Layers	Attack Name	Attack References	Effects	Launch	Countermeasure
Perception Layer	Hardware Tempering	[8]	Data leakage (Keys, routing tables, etc)	2011	Secure Physical Design
	Fake node injection	[7]	Fake Data Manipulation	2013	Secure Booting
	Malicious code injection	[9]	Halt Transmission	2012	Intrusion detection Technology(IDT)
	Sleep denial attack	[10]	Node shutdown	2012	Authentication
	WSN Node Jamming	[11]	Jam Node Communication	2010	IPSec Security channel
	RF interference of RFIDs	[12]	Distortion in node Communication	2012	Authentication
Network Layer	Traffic analysis attack	[19]	Data leakage (about network)	2013	Routing Security
	RFID Spoofing	[20]	Intrusion in network Data manipulation	2011	GPS Location System
	RFID unauthorized access	[21]	Node data can be modified (Read, Write & Delete)	2014	Network Authentication
	Sinkhole Attack	[22]	Data leakage (Data of the Nodes)	2013	Security Aware AdHoc Routing
	Man in the Middle Attack	[23]	Data Privacy Violation	2011	Point-to-Point Encryption
	Routing Information Attack	[24]	Routing loops (Network Destruction)	2011	Encrypting Routing Tables
Processing Layer	Application security	[30]	Privacy Violation	2014	Web Application Scanner
	Data security	[31]	Data leakage (User data on cloud)	2012	Homomorphic Encryption
	Underlying infrastructure security	[32]	Service Hijacking	2010	Fragmentation redundancy scattering
	Third-party relationships	[33]	Data Leakage (User data on cloud)	2013	Encryption
	Virtualization threats	[34]	Resources destruction	2012	Hyper Safe
	Shared Resources	[35]	Resources Theft	2011	Hyper Safe
Application Layer	Phishing Attacks	[40]	Data Leakage (User credentials data)	2016	Biometrics Authentication
	Virus, Worms, Trojan Horse, Spyware	[41]	Resource Destruction & Hijacking	2012	Protective Software
	Malicious Scripts	[42]	Hijacking	2011	Firewalls
	Denial of Service(DoS)	[43]	Resource Destruction	2010	Access Control Lists
	Data Protection and Recovery	[44]	Data loss & Catastrophic Damage	2011	Cryptographic Hash Functions
	Software Vulnerabilities	[45]	Buffer over flow	2011	Awareness of security

Figure C.1: Summary of IoT communication protocols [7]

.

Appendix D

Robot Photos

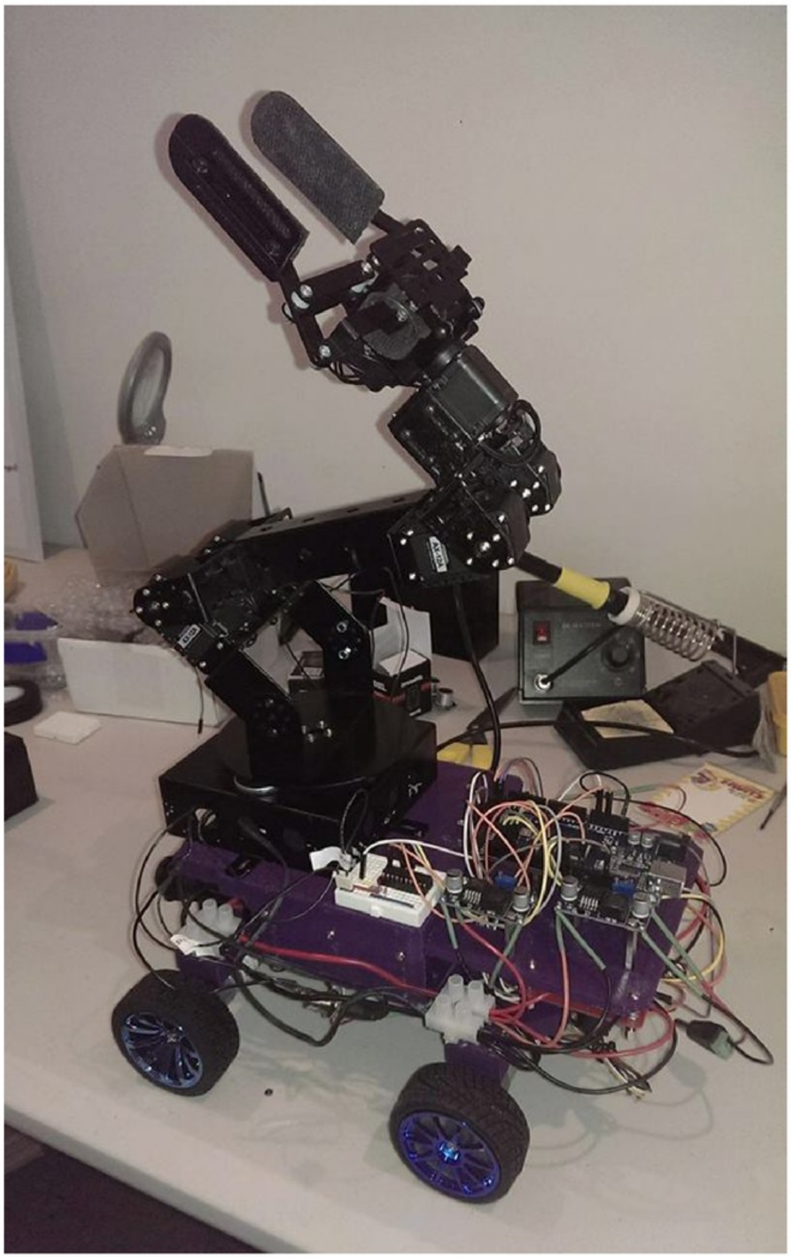


Figure D.1: Robot with arm attached.

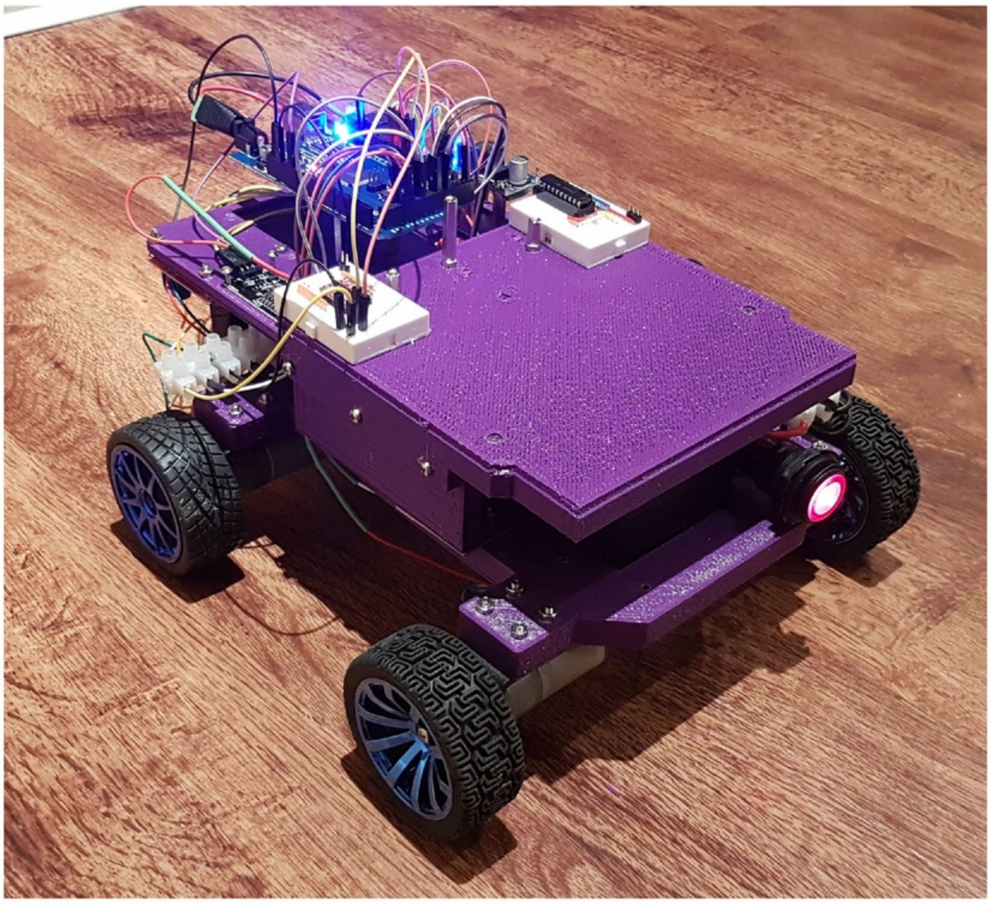


Figure D.2: Robot without arm attached.

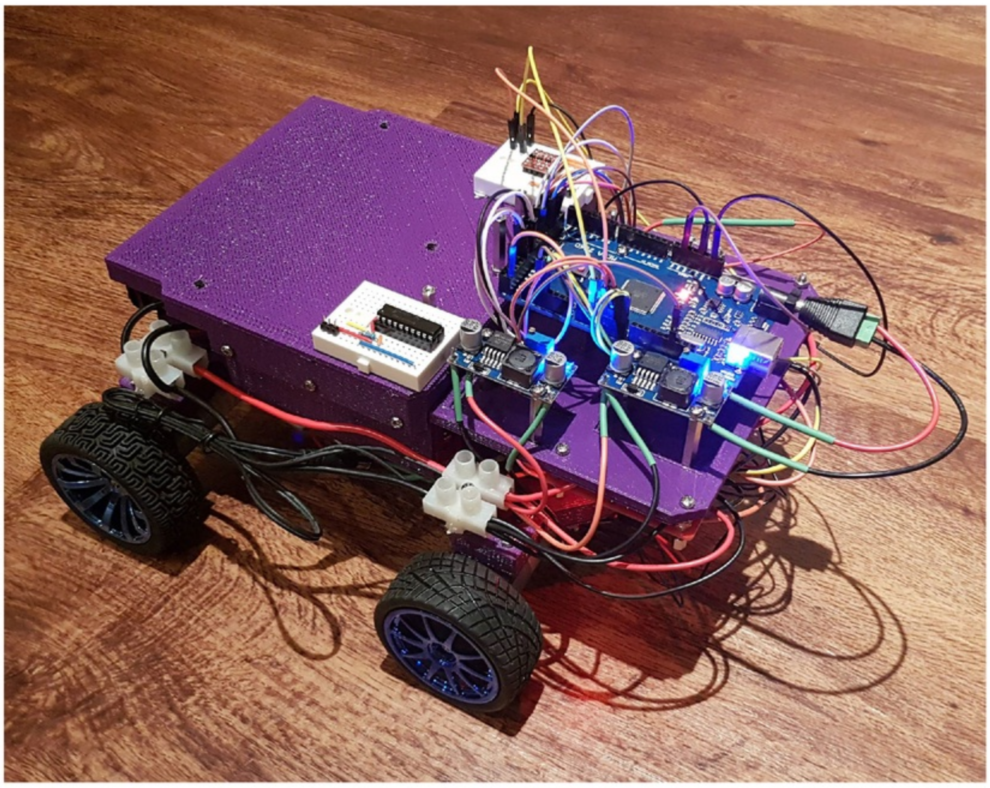


Figure D.3: Robot without arm attached.

Appendix E

Web Server Code

E.1 app.js

```
var express = require('express');
var path = require('path');
var favicon = require('serve-favicon');
var logger = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');
var mysql = require('mysql');
var debug = require('debug')('thesis-server:server');
var http = require('http');

var index = require('./routes/index');
var users = require('./routes/users');

var app = express();

var con = mysql.createConnection({
  host: "localhost",
  user: "user",
  password: "password",
  database: "thesisDB"
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  var sql = "CREATE TABLE thesisTable (id INT AUTO_INCREMENT
     PRIMARY KEY, address VARCHAR(255), timestamp VARCHAR(255),
     speed VARCHAR(255),angle VARCHAR(255),power VARCHAR(255),
     encoder1 VARCHAR(255),cmdreceived VARCHAR(255))";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Table created");
  });
});

// Global variable setup (not ideal but used for functional
   testing)
global.speedbar = 0;
global.turnbar = 0;
global.LED = 0;
global.power = 0;
global.newCommand = 0;
global.servos = "150150150150150";

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');

app.set('port', (8888));
app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

// Setup Routes
  var arduinoComm = require('./routes/arduinoComm');
  var androidComm = require('./routes/androidComm');

  app.use ('/arduino', arduinoComm);
  app.use ('/android', androidComm);
  app.use('/', index);

// Initialise server and listen for connections
app.listen(app.get('port'), function() {
  console.log('Thesis webserver listening on port: ' + app.get('
     port'));
});
```

E.2 arduinoComm.js

```
var express = require('express');
var router = express.Router();
var bodyParser = require('body-parser');
var querystring = require('querystring');
var mysql = require('mysql');
var url = require('url');
var crypto = require("crypto");

var currentID = 0;
var key = new Buffer( '5B5A57676A56676E', 'hex' );
var iv = new Buffer(8);
iv.fill(0);
var outputData;

var con = mysql.createConnection({
  host: "localhost",
  user: "user",
  password: "password",
  database: "thesisDB"
});

router.use(bodyParser.text({ type: 'text/html' }));

router.get ('/', function(req, res) {
  sqlInsert2(req,dataString);

  speedbar = 0;
  turnbar = 0;
  res.write(outputData);
  res.end();
});

function sqlInsert2(req,callback){
  con.connect(function(err) {
    if (err) throw err;
    console.log("Connected!");
    var sql = "UPDATE thesisTable SET encoder1=req.query.
       encoder1, cmdreceived=1 WHERE ID = currentID";
    con.query(sql, function (err, result) {
     if (err) throw err;
     console.log("1 record inserted");
```

```
      currentID+=1;
    });
  });
  callback(encryptDES);
}

function dataString(callback) {
  var returnString = "?LED=";
  returnString += LED;
  returnString += "&speedbar=";
  returnString += speedbar;
  returnString += "&turnbar=";
  returnString += turnbar;
  returnString += "&power=";
  returnString += power;
  returnString += "&servos=";
  returnString += servos;
  returnString += " ";

  var len = returnString.length;
  var padding = 64-len;
  for(var i = 0;i<padding;i++){
    returnString += " ";
  }
  callback(returnString);
}

function encryptDES(plaintext){
    outputData="";
    for(var i = 0;i<64;i+=8){
        var subStr = plaintext.substring(i,i+8);
        var cipher = crypto.createCipheriv("des", key,iv);
        cipher.setAutoPadding(false);
        var c = cipher.update( subStr, 'binary', 'hex' );
        c+=cipher.final('hex' );
        outputData += c;
    }
}

module.exports = router;
```

E.3 androidComm.js

```
var express = require('express');
var router = express.Router();
var bodyParser = require('body-parser');
var mysql = require('mysql');
var querystring = require('querystring');

var con = mysql.createConnection({
  host: "localhost",
  user: "user",
  password: "password",
  database: "thesisDB"
});

router.post ('/', function(req, res) {
    saveVariables(req,sqlInsert);
    res.send('Command Sent');
});

router.get ('/', function(req, res) {
    if(req.query.conTest == 1){
        res.send('Success');
    }
    else{
        res.send('testFail');
    }
});

function saveVariables(req,callback){
    var address = req.connection.remoteAddress;
    var timeStart = req.connection.timeStart;
    speedbar = req.body.speedbar;
    turnbar = req.body.turnbar;
    LED = req.body.LED;
    power = req.body.power;
    servos = req.body.servos;

    callback(address,timeStart);
}

function sqlInsert(addr,time){
  con.connect(function(err) {
    if (err) throw err;
    console.log("Connected!");
    var sql = "INSERT INTO thesisTable (address, timestamp,
       speed, angle, power) VALUES (addr, timestart, speedbar,
       turnbar, power)";
    con.query(sql, function (err, result) {
     if (err) throw err;
     console.log("1 record inserted");
    });
 });
}
module.exports = router;
```

Appendix F

Android Application Code

F.1 MainActivity.java

```
package com.vark.aard.thesis_app;


import android.os.Bundle;
import android.support.design.widget.TabLayout;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;


public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        TabLayout tabLayout = (TabLayout) findViewById(R.id.
           tab_layout);
        tabLayout.addTab(tabLayout.newTab().setText("Settings"))
           ;
        tabLayout.addTab(tabLayout.newTab().setText("Movement
            Control"));
        tabLayout.addTab(tabLayout.newTab().setText("Robot Arm
            Control"));
        tabLayout.setTabGravity(TabLayout.GRAVITY_FILL);

        final ViewPager viewPager = (ViewPager) findViewById(R.
            id.pager);
        final PagerAdapter adapter = new PagerAdapter
                (getSupportFragmentManager(), tabLayout.
                    getTabCount());
        viewPager.setAdapter(adapter);
        viewPager.addOnPageChangeListener(new TabLayout.
            TabLayoutOnPageChangeListener(tabLayout));
        tabLayout.addOnTabSelectedListener(new TabLayout.
            OnTabSelectedListener() {
             @Override
             public void onTabSelected(TabLayout.Tab tab) {
                 viewPager.setCurrentItem(tab.getPosition());
             }

             @Override
             public void onTabUnselected(TabLayout.Tab tab) {

             }

             @Override
             public void onTabReselected(TabLayout.Tab tab) {

             }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

F.2 PagerAdapter.java

```
package com.vark.aard.thesis_app;

/**
 * Created by David on 4/09/2017.
 */


import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentStatePagerAdapter;

public class PagerAdapter extends FragmentStatePagerAdapter {
    int mNumOfTabs;

    public PagerAdapter(FragmentManager fm, int NumOfTabs) {
        super(fm);
        this.mNumOfTabs = NumOfTabs;
    }

    @Override
    public Fragment getItem(int position) {

        switch (position) {
            case 0:
                TabFragment1 tab1 = new TabFragment1();
                return tab1;
            case 1:
                TabFragment2 tab2 = new TabFragment2();
                return tab2;
            case 2:
                TabFragment3 tab3 = new TabFragment3();
                return tab3;
            default:
                return null;
        }
    }

    @Override
    public int getCount() {
        return mNumOfTabs;
    }
}
```

F.3 SingletonGlobal.java

```
package com.vark.aard.thesis_app;

/**
 * Created by David on 4/09/2017.
 */

public class SingletonGlobal {
    private static final SingletonGlobal ourInstance = new
       SingletonGlobal();
    private String mSpeedBar = "0";
    private String mTurnBar = "0";
    private String mCommandReceived = "0";
    private String mLED = "0";
    private String mPower = "1";
    private Boolean mCommandSent = false;
    private String mRobotAngles = "150150150150150";

    public static SingletonGlobal getInstance() {
        return ourInstance;
    }

    private SingletonGlobal() {
    }

    public String getSpeedbarVal() {
        return this.mSpeedBar;
    }

    public void setSpeedbarVal(String str) {
        this.mSpeedBar = str;
    }

    public String getTurnbarVal() {
        return this.mTurnBar;
    }

    public void setTurnbarVal(String str) {
        this.mTurnBar = str;
    }

    public String getCommandReceived(){ return this.
```

```
        mCommandReceived;  }

    public void setCommandReceived(String str){ this.
        mCommandReceived = str;}

    public String getLEDStatus(){ return this.mLED;}

    public void setLEDStatus(String str){ this.mLED = str;}

    public String getPowerStatus(){ return this.mPower;}

    public void setPowerStatus(String str){ this.mPower = str;}

    public Boolean getCommandSent(){return this.mCommandSent;}

    public void setCommandSent(Boolean bool){this.mCommandSent =
         bool;}

    public void setRobotAngles(String str){ this.mRobotAngles =
        str;}

    public String getRobotAngles(){ return this.mRobotAngles;}
}
```

F.4 httpTransfer.java

```
package com.vark.aard.thesis_app;

import android.content.ContentValues;
import android.content.Context;
import android.content.SharedPreferences;
import android.os.AsyncTask;
import android.os.Handler;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.TextView;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;

/**
 * Created by David on 16/07/2017.
 */

public class httpTransfer extends AsyncTask<Context, Integer,
   String> {
    SharedPreferences prefs;
    SharedPreferences.Editor editor;
    private Context mContext;
    private View mView;
    private TextView textView14,textView15,textView24;
    private Integer tabInt;

    public httpTransfer(Context passedContext, View passedView,
       Integer tabNumber){
        mContext = passedContext;
        mView = passedView;
        tabInt = tabNumber;
        if(tabInt==1) {
            textView15 = (TextView) mView.findViewById(R.id.
               textView15);
        }
        else if (tabInt==2){
            textView14 = (TextView) mView.findViewById(R.id.
               textView14);
        }
        else if (tabInt==3){
            textView14 = (TextView) mView.findViewById(R.id.
               textView14);
        }
        else if (tabInt==4){
            textView24 = (TextView) mView.findViewById(R.id.
               textView24);
        }
        else{
            //how did you get here?!
        }
    }

    @Override
    protected void onPreExecute(){
        super.onPreExecute();
        if(tabInt==1) {
            textView15.setBackgroundResource(R.drawable.rect);
            textView15.setText("Attempting Connection");
        }
        else if (tabInt==2){
            textView14.setBackgroundResource(R.drawable.rect);
            textView14.setText("Attempting Connection");
        }
        else if (tabInt==3){
            //no changes as waiting for get request to check if
               commands received
        }
        else if (tabInt==4){
            textView24.setBackgroundResource(R.drawable.rect);
            textView24.setText("Attempting Connection");
        }
        else{
            //Huh?
        }
```

```
}

@Override
protected String doInBackground(Context... params){
    prefs = mContext.getSharedPreferences("Connection",
       Context.MODE_PRIVATE);
    editor = prefs.edit();
    String result;
    if(tabInt==1) {
        result = sendGetRequest();
    }
    else if (tabInt==2){
        result = sendPostRequest();
    }
    else if(tabInt==3){
        result = sendGetRequest();
    }
    else if (tabInt==4){
        result = sendPostRequest();
    }
    else{ result = null; }

    return result;
}

protected void onPostExecute(String getResult){
    super.onPostExecute(getResult);
    if(tabInt==1) {
        if(TextUtils.isEmpty( getResult )){
            textView15.setBackgroundResource(R.drawable.
               buttonrect3);
            textView15.setText("Connection Failed");
        }
        else if(getResult.equals("Success")){
            textView15.setBackgroundResource(R.drawable.
               buttonrect2);
            textView15.setText("Connection OK");
        }
        else{
            textView15.setBackgroundResource(R.drawable.
               buttonrect3);
            textView15.setText("Connection Failed");
        }
        }
        else if (tabInt==2){
            if(TextUtils.isEmpty( getResult )){
                textView14.setBackgroundResource(R.drawable.
                   buttonrect3);
                textView14.setText("Connection Failed");
            }
            else if(getResult.equals("Command Sent")){
                textView14.setBackgroundResource(R.drawable.
                   buttonrect2);
                textView14.setText("Connection OK");
                SingletonGlobal.getInstance().setCommandSent(
                   false);
            }
            else{
                textView14.setBackgroundResource(R.drawable.
                   buttonrect3);
                textView14.setText("Connection Failed");
            }

        }
        else if (tabInt==3){

        }
        else if(tabInt==4) {
            if(TextUtils.isEmpty( getResult )){
                textView24.setBackgroundResource(R.drawable.
                   buttonrect3);
                textView24.setText("Connection Failed");
            }
            else if(getResult.equals("Command Sent")){
                textView24.setBackgroundResource(R.drawable.
                   buttonrect2);
                textView24.setText("Connection OK");
            }
            else{
                textView24.setBackgroundResource(R.drawable.
                   buttonrect3);
                textView24.setText("Connection Failed");
            }
        }
        else{ }
    }
```

```
public String sendPostRequest(){
    String inputLine;
    String result = "";
    StringBuilder stringURLbuild = new StringBuilder();
    stringURLbuild.append("http://");
    stringURLbuild.append(prefs.getString("ipAddress
       ","192.168.1.12"));
    stringURLbuild.append(":");
    stringURLbuild.append(prefs.getString("port","8888"));
    stringURLbuild.append("/android");
    String stringUrl = stringURLbuild.toString();

    HttpURLConnection connection = null;
    try {
        //Create a URL object holding our url
        URL myUrl = new URL(stringUrl);

        //Create a connection
        connection =(HttpURLConnection)
                myUrl.openConnection();

        //Set methods and timeouts
        connection.setRequestMethod("POST");
        connection.setReadTimeout(4000);
        connection.setConnectTimeout(4000);
        connection.setDoOutput(true);

        ContentValues values = new ContentValues();
        values.put("portNum",prefs.getString("port","8888"))
            ;
        if(SingletonGlobal.getInstance().getPowerStatus().
           equals("0")){
            values.put("speedbar","-1");
            values.put("turnbar","0");
        }
        else{
            values.put("speedbar",SingletonGlobal.
               getInstance().getSpeedbarVal());
            values.put("turnbar",SingletonGlobal.getInstance
               ().getTurnbarVal());
        }
```

```
values.put("newCommand","1");
values.put("LED",SingletonGlobal.getInstance().
   getLEDStatus());
values.put("power",SingletonGlobal.getInstance().
   getPowerStatus());
values.put("servos",SingletonGlobal.getInstance().
   getRobotAngles());


StringBuilder valueSB = new StringBuilder();
for (String name : values.keySet()) {
    valueSB.append(URLEncoder.encode(name,"UTF-8"));
    valueSB.append("=");
    valueSB.append(URLEncoder.encode(values.
       getAsString(name),"UTF-8"));
    valueSB.append("&");
}
valueSB.setLength(valueSB.length()-1);

OutputStream out = connection.getOutputStream();
BufferedWriter writer = new BufferedWriter(new
   OutputStreamWriter(out, "UTF-8"));
writer.write(valueSB.toString());
writer.flush();
writer.close();
out.close();

//Connect to our url
connection.connect();
 if(connection.getInputStream()!=null) {
    //Create a new InputStreamReader
    InputStreamReader streamReader = new
            InputStreamReader(connection.
               getInputStream());
    //Create a new buffered reader and String
       Builder
    BufferedReader reader = new BufferedReader(
       streamReader);
    StringBuilder stringBuilder = new StringBuilder
       ();
    //Check if the line we are reading is not null
    while ((inputLine = reader.readLine()) != null)
```

```
                {
                  stringBuilder.append(inputLine);
              }
              //Close our InputStream and Buffered reader
              reader.close();
              streamReader.close();
              //Set our result equal to our stringBuilder
              result = stringBuilder.toString();
          }
          else{  }
      } catch(IOException e) {
          e.printStackTrace();
          result = null;
      } finally {
          connection.disconnect();
      }

      return result;
  }

  public String sendGetRequest(){
      String inputLine;
      String getResult = "";
      StringBuilder stringURLbuild = new StringBuilder();
      stringURLbuild.append("http://");
      stringURLbuild.append(prefs.getString("ipAddress
         ","192.168.1.12"));
      stringURLbuild.append(":");
      stringURLbuild.append(prefs.getString("port","8888"));
      stringURLbuild.append("/android");

      stringURLbuild.append("?");
      if(tabInt==1) {
          stringURLbuild.append("conTest=1");
      }
      else if (tabInt==2 || tabInt==3){
          stringURLbuild.append("conTest=0");
      }

      String stringUrl = stringURLbuild.toString();
      HttpURLConnection connection = null;
      try {
          //Create a URL object holding our url
```

```
            URL myUrl = new URL(stringUrl);

            //Create a connection
            connection =(HttpURLConnection)
                    myUrl.openConnection();

            //Set methods and timeouts
            connection.setRequestMethod("GET");
            connection.setReadTimeout(8000);
            connection.setConnectTimeout(8000);

            //Connect to our url
            connection.connect();
            if(connection.getInputStream()!=null) {
                //Create a new InputStreamReader
                InputStreamReader streamReader = new
                        InputStreamReader(connection.
                           getInputStream());
                //Create a new buffered reader and String
                   Builder
                BufferedReader reader = new BufferedReader(
                   streamReader);
                StringBuilder stringBuilder = new StringBuilder
                   ();
                //Check if the line we are reading is not null
                while ((inputLine = reader.readLine()) != null)
                   {
                    stringBuilder.append(inputLine);
                }
                //Close our InputStream and Buffered reader
                reader.close();
                streamReader.close();
                //Set our result equal to our stringBuilder
                getResult = stringBuilder.toString();
            }
            else{}

        } catch(IOException e) {
            e.printStackTrace();
            getResult = null;
        } finally {
            connection.disconnect();
        }
        return getResult;
    }
}
```

F.5 TabFragment1.java

```
package com.vark.aard.thesis_app;

/**
 * Created by David on 4/09/2017.
 */

import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.text.TextUtils;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import java.util.concurrent.ExecutionException;

public class TabFragment1 extends Fragment {


    private EditText editTextIPAddress, editTextPortNumber;
    private Button portButton,ipButton,testButton;
    private TextView textView13,textView15;
    private static final String TAG = "MainActivity";
    SharedPreferences prefs;
    SharedPreferences.Editor editor;

    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        prefs = getActivity().getSharedPreferences("Connection",
           Context.MODE_PRIVATE);
        editor = prefs.edit();
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
 final View viewtab1 = inflater.inflate(R.layout.
    tab_fragment_1, container, false);
 editTextIPAddress = (EditText)viewtab1.findViewById(R.id
    .editText1);
 editTextPortNumber = (EditText)viewtab1.findViewById(R.
    id.editText2);
 portButton = (Button)viewtab1.findViewById(R.id.IPButton
    );
 ipButton = (Button)viewtab1.findViewById(R.id.PortButton
    );
 testButton = (Button)viewtab1.findViewById(R.id.
    TestButton);
 textView13 = (TextView)viewtab1.findViewById(R.id.
    textView13);
 textView15 = (TextView)viewtab1.findViewById(R.id.
    textView15);
 setConnectText();


 portButton.setOnClickListener(new View.OnClickListener()
     {
     @Override
     public void onClick(View v) {
          String newIP = editTextIPAddress.getText().
             toString();
          editor.putString("ipAddress", newIP);
          editor.commit();
          Log.d(TAG,"IP address button");
          Log.d(TAG,newIP);
          setConnectText();
     }
 });

 ipButton.setOnClickListener(new View.OnClickListener() {
     @Override
     public void onClick(View v) {
          String newPort = editTextPortNumber.getText().
             toString();
          editor.putString("port", newPort);
          editor.commit();
          Log.d(TAG,"Port number button");
          Log.d(TAG,newPort);
```

```
                setConnectText();
            }
        });

        testButton.setOnClickListener(new View.OnClickListener()
            {
            @Override
            public void onClick(View v) {
                new httpTransfer(getContext(),viewtab1,1).
                    execute();
            }
        });

        return viewtab1;
    }

    public void setConnectText(){
        StringBuilder sb = new StringBuilder();
        sb.append("http://");
        sb.append(prefs.getString("ipAddress","192.168.1.12"));
        sb.append(":");
        sb.append(prefs.getString("port","8888"));
        String currentConnect = sb.toString();
        textView13.setText(currentConnect);
    }
}
```

F.6 TabFragment2.java

```
package com.vark.aard.thesis_app;

/**
 * Created by David on 4/09/2017.
 */

import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.os.Handler;
import android.support.v4.app.Fragment;
import android.text.TextUtils;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.TextView;

import java.util.concurrent.ExecutionException;

public class TabFragment2 extends Fragment {

    private Button cmdButton, ledButton, pwrButton, fwdButton,
       leftButton,rightButton,bwdButton,stopButton;
    private TextView textView14,textView2,textView3;
    SharedPreferences prefs;
    SharedPreferences.Editor editor;
    private View viewtab2;
    private Integer fwdCount = 0;
    private Integer turnCount = 0;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        prefs = getActivity().getSharedPreferences("Connection",
            Context.MODE_PRIVATE);
        editor = prefs.edit();
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
       container, Bundle savedInstanceState) {
        viewtab2 = inflater.inflate(R.layout.tab_fragment_2,
           container, false);
        cmdButton = (Button) viewtab2.findViewById(R.id.
           commandButton);
        ledButton = (Button) viewtab2.findViewById(R.id.
           LEDButton);
        textView14 = (TextView) viewtab2.findViewById(R.id.
           textView14);
        textView2 = (TextView) viewtab2.findViewById(R.id.
           textView2);
        textView3 = (TextView) viewtab2.findViewById(R.id.
           textView3);
        pwrButton = (Button) viewtab2.findViewById(R.id.
           powerButton);
        fwdButton = (Button) viewtab2.findViewById(R.id.
           fwdButton);
        leftButton = (Button) viewtab2.findViewById(R.id.
           leftButton);
        rightButton = (Button) viewtab2.findViewById(R.id.
           rightButton);
        bwdButton = (Button) viewtab2.findViewById(R.id.
           bwdButton);
        stopButton = (Button) viewtab2.findViewById(R.id.
           stopButton);


        cmdButton.setOnClickListener(new View.OnClickListener()
           {
            @Override
            public void onClick(View v) {
                if (SingletonGlobal.getInstance().getCommandSent
                   ().equals(false)) {
                    SingletonGlobal.getInstance().setCommandSent
                       (true);
                    SingletonGlobal.getInstance().
                       setCommandReceived("0");
                    new httpTransfer(getContext(), viewtab2, 2).
                       execute();
                    textView2.setText("0");
                    textView3.setText("0");
                fwdCount = 0;
            }

        }
    });
    ledButton.setOnClickListener(new View.OnClickListener()
       {
        @Override
        public void onClick(View v) {
            if (SingletonGlobal.getInstance().getLEDStatus()
               .equals("0")) {
                SingletonGlobal.getInstance().setLEDStatus
                   ("1");
            } else {
                SingletonGlobal.getInstance().setLEDStatus
                   ("0");
            }
        }
    });

    pwrButton.setOnClickListener(new View.OnClickListener()
       {
        @Override
        public void onClick(View v) {
            if (SingletonGlobal.getInstance().getPowerStatus
               ().equals("0")) {
                SingletonGlobal.getInstance().setPowerStatus
                   ("1");
            } else {
                SingletonGlobal.getInstance().setPowerStatus
                   ("0");

            }
        }
    });

    fwdButton.setOnClickListener(new View.OnClickListener()
       {
        @Override
        public void onClick(View v) {
            fwdCount += 100;
            SingletonGlobal.getInstance().setSpeedbarVal(
               fwdCount.toString());
            textView2.setText(fwdCount.toString());
        }
    });

    bwdButton.setOnClickListener(new View.OnClickListener()
        {
         @Override
         public void onClick(View v) {
             fwdCount -= 100;
             SingletonGlobal.getInstance().setSpeedbarVal(
                fwdCount.toString());
             textView2.setText(fwdCount.toString());
         }
    });

    stopButton.setOnClickListener(new View.OnClickListener()
         {
         @Override
         public void onClick(View v) {
             SingletonGlobal.getInstance().setSpeedbarVal
                ("-1");
                             new httpTransfer(getContext(),
                                viewtab2, 2).execute();
             textView2.setText("0");
         }
    });

    leftButton.setOnClickListener(new View.OnClickListener()
         {
         @Override
         public void onClick(View v) {
             turnCount -= 1;
             SingletonGlobal.getInstance().setTurnbarVal(
                turnCount.toString());
             Integer temp = 45*turnCount;
             textView3.setText(temp.toString());
         }
    });

    rightButton.setOnClickListener(new View.OnClickListener
        () {
         @Override
         public void onClick(View v) {
                turnCount += 1;
                SingletonGlobal.getInstance().setTurnbarVal(
                   turnCount.toString());
                Integer temp = 45*turnCount;
                textView3.setText(temp.toString());
            }
        });

        return viewtab2;
    }

}
```

F.7 TabFragment3.java

```
package com.vark.aard.thesis_app;

/**
 * Created by David on 4/09/2017.
 */

import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.TextView;

public class TabFragment3 extends Fragment {

    private TextView textView11,textView17,textView19,textView21
       ,textView23,textView24;
    private SeekBar seekBar3,seekBar4,seekBar5,seekBar6,seekBar7
       ;
    private Button resetAngle,robotSend;
    SharedPreferences prefs;
    SharedPreferences.Editor editor;
    private View viewtab3;

    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        prefs = getActivity().getSharedPreferences("Connection",
            Context.MODE_PRIVATE);
        editor = prefs.edit();
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
       container, Bundle savedInstanceState) {
        viewtab3 = inflater.inflate(R.layout.tab_fragment_3,
        container, false);
    textView11 = (TextView) viewtab3.findViewById(R.id.
        textView11);
    textView17 = (TextView) viewtab3.findViewById(R.id.
        textView17);
    textView19 = (TextView) viewtab3.findViewById(R.id.
        textView19);
    textView21 = (TextView) viewtab3.findViewById(R.id.
        textView21);
    textView23 = (TextView) viewtab3.findViewById(R.id.
        textView23);
    textView24 = (TextView) viewtab3.findViewById(R.id.
        textView24);
    seekBar3 = (SeekBar) viewtab3.findViewById(R.id.seekBar3
        );
    seekBar4 = (SeekBar) viewtab3.findViewById(R.id.seekBar4
        );
    seekBar5 = (SeekBar) viewtab3.findViewById(R.id.seekBar5
        );
    seekBar6 = (SeekBar) viewtab3.findViewById(R.id.seekBar6
        );
    seekBar7 = (SeekBar) viewtab3.findViewById(R.id.seekBar7
        );
    resetAngle = (Button) viewtab3.findViewById(R.id.
        resetAngle);
    robotSend = (Button) viewtab3.findViewById(R.id.
        robotSend);
    setAngleText();

    seekBar3.setOnSeekBarChangeListener(new SeekBar.
        OnSeekBarChangeListener() {
         @Override
         public void onProgressChanged(SeekBar seekBar, int
             progress,
                                       boolean fromUser) {
             int seekValue = seekBar.getProgress();
             textView11.setText(String.valueOf(seekValue));
         }

         @Override
         public void onStartTrackingTouch(SeekBar seekBar) {
             // TODO Auto-generated method stub
         }
```

```
            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });

        seekBar4.setOnSeekBarChangeListener(new SeekBar.
           OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int
               progress,
                                          boolean fromUser) {
                int seekValue = seekBar.getProgress();
                textView17.setText(String.valueOf(seekValue));
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
                // TODO Auto-generated method stub
            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });

        seekBar5.setOnSeekBarChangeListener(new SeekBar.
           OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int
               progress,
                                          boolean fromUser) {
                int seekValue = seekBar.getProgress();
                textView19.setText(String.valueOf(seekValue));
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
                // TODO Auto-generated method stub
            }
            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });

        seekBar6.setOnSeekBarChangeListener(new SeekBar.
           OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int
               progress,
                                           boolean fromUser) {
                int seekValue = seekBar.getProgress();
                textView21.setText(String.valueOf(seekValue));
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
                // TODO Auto-generated method stub
            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });

        seekBar7.setOnSeekBarChangeListener(new SeekBar.
           OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int
               progress,
                                           boolean fromUser) {
                int seekValue = seekBar.getProgress();
                textView23.setText(String.valueOf(seekValue));
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
                // TODO Auto-generated method stub
            }

            @Override
        public void onStopTrackingTouch(SeekBar seekBar) {

        }
    });

    resetAngle.setOnClickListener(new View.OnClickListener()
        {
        @Override
        public void onClick(View v) {
            seekBar3.setProgress(150);
            seekBar4.setProgress(150);
            seekBar5.setProgress(150);
            seekBar6.setProgress(150);
            seekBar7.setProgress(150);
            String resetAngles = "150150150150150";
            SingletonGlobal.getInstance().setRobotAngles(
               resetAngles);
        }
    });

    robotSend.setOnClickListener(new View.OnClickListener()
       {
        @Override
        public void onClick(View v) {
            StringBuilder robotString = new StringBuilder();
            if(seekBar3.getProgress()<10) {
                robotString.append("00");
                robotString.append(seekBar3.getProgress());
            }
            else if(seekBar3.getProgress()<100){
                robotString.append("0");
                robotString.append(seekBar3.getProgress());
            }
            else{
                robotString.append(seekBar3.getProgress());
            }
            if(seekBar4.getProgress()<10) {
                robotString.append("00");
                robotString.append(seekBar4.getProgress());
            }
            else if(seekBar4.getProgress()<100){
                robotString.append("0");
                robotString.append(seekBar4.getProgress());
```

```
}
else{
    robotString.append(seekBar4.getProgress());
}
if(seekBar5.getProgress()<10) {
    robotString.append("00");
    robotString.append(seekBar5.getProgress());
}
else  if(seekBar5.getProgress()<100){
    robotString.append("0");
    robotString.append(seekBar5.getProgress());
}
else{
    robotString.append(seekBar5.getProgress());
}
if(seekBar6.getProgress()<10) {
    robotString.append("00");
    robotString.append(seekBar6.getProgress());
}
else  if(seekBar6.getProgress()<100){
    robotString.append("0");
    robotString.append(seekBar6.getProgress());
}
else{
    robotString.append(seekBar6.getProgress());
}
if(seekBar7.getProgress()<10) {
    robotString.append("00");
    robotString.append(seekBar7.getProgress());
}
else  if(seekBar7.getProgress()<100){
    robotString.append("0");
    robotString.append(seekBar7.getProgress());
}
else{
    robotString.append(seekBar7.getProgress());
}

String robotSendString = robotString.toString();
SingletonGlobal.getInstance().setRobotAngles(
   robotSendString);
new httpTransfer(getContext(),viewtab3,4).
   execute();
```

```
            }
        });

        return viewtab3;
    }

    public void setAngleText(){
        textView11.setText("150");
        textView17.setText("150");
        textView19.setText("150");
        textView21.setText("150");
        textView23.setText("150");
    }
}
```

Appendix G

Arduino Code

```
/*
 * Robotic System Control and Wireless Communication Platform
 * Thesis Project: David Simpson - 2017
 *
 * HOST_NAME = IP Address of webserver
 * (192.168.1.12 for local network testing)
 * (14.202.147.3 for proper operation and internet access)
 *
 * !This IP address may change if updated by ISP as webserver is
 *  hosted on home network!
 *
 *  Encoder parts altered from http://www.hessmer.org/blog
    /2011/01/30/quadrature-encoder-too-fast-for-arduino-with-
    solution/
 *  DES encryption algorithm derived from http://arduino-
    projects4u.com/des-algorithm/
 */

//Include library files
#include "ESP8266.h"
#include <DynamixelSerial2.h>
#include <digitalWriteFast.h>
#include <stdint.h>
#include <string.h>

//Define program constants
#define SSID          "Network Name"
#define PASSWORD      "password"

//#define HOST_NAME    "192.168.1.12"        //local testing
```

```
   address
#define HOST_NAME    "14.202.147.3"     //internet IP address
#define HOST_PORT    (8888)
#define DEBUG        false

//Initialise ESP8266 WiFi device to communicate on Serial1 port
   (BAUD rate for my ESP module is 115200)
ESP8266 wifi(Serial1);

/*---------- Variables ----------*/
int speedValue = 0;
int turnValue = 0;
int LEDStatus = 0;
int powerStatus = 0;
int newCommand = 0;
int cmdReceived = 0;
unsigned long previousMillis = 0;
unsigned long interval = 2500;
int baseAngle = 512;
int shoulderAngle = 512;
int elbowAngle = 512;
int wristAngle = 512;
int gripAngle = 512;
long L_Target = 0;
long R_Target = 0;
long Target = 0;
boolean fortyfiveFlag = false;
boolean moveFlag = false;
boolean lastCmdCompleted = true;
char directionChar = 'f';
uint8_t buffer2[1024];
uint32_t cmdBuffer[64];
/*-------------------------------*/

/*------ Encryption Constants --------*/
const uint8_t sbox[256]  = {
  /* S-box 1 */
  0xE4, 0xD1, 0x2F, 0xB8, 0x3A, 0x6C, 0x59, 0x07,
  0x0F, 0x74, 0xE2, 0xD1, 0xA6, 0xCB, 0x95, 0x38,
  0x41, 0xE8, 0xD6, 0x2B, 0xFC, 0x97, 0x3A, 0x50,
  0xFC, 0x82, 0x49, 0x17, 0x5B, 0x3E, 0xA0, 0x6D,
  /* S-box 2 */
  0xF1, 0x8E, 0x6B, 0x34, 0x97, 0x2D, 0xC0, 0x5A,
  0x3D, 0x47, 0xF2, 0x8E, 0xC0, 0x1A, 0x69, 0xB5,
  0x0E, 0x7B, 0xA4, 0xD1, 0x58, 0xC6, 0x93, 0x2F,
  0xD8, 0xA1, 0x3F, 0x42, 0xB6, 0x7C, 0x05, 0xE9,
  /* S-box 3 */
  0xA0, 0x9E, 0x63, 0xF5, 0x1D, 0xC7, 0xB4, 0x28,
  0xD7, 0x09, 0x34, 0x6A, 0x28, 0x5E, 0xCB, 0xF1,
  0xD6, 0x49, 0x8F, 0x30, 0xB1, 0x2C, 0x5A, 0xE7,
  0x1A, 0xD0, 0x69, 0x87, 0x4F, 0xE3, 0xB5, 0x2C,
  /* S-box 4 */
  0x7D, 0xE3, 0x06, 0x9A, 0x12, 0x85, 0xBC, 0x4F,
  0xD8, 0xB5, 0x6F, 0x03, 0x47, 0x2C, 0x1A, 0xE9,
  0xA6, 0x90, 0xCB, 0x7D, 0xF1, 0x3E, 0x52, 0x84,
  0x3F, 0x06, 0xA1, 0xD8, 0x94, 0x5B, 0xC7, 0x2E,
  /* S-box 5 */
  0x2C, 0x41, 0x7A, 0xB6, 0x85, 0x3F, 0xD0, 0xE9,
  0xEB, 0x2C, 0x47, 0xD1, 0x50, 0xFA, 0x39, 0x86,
  0x42, 0x1B, 0xAD, 0x78, 0xF9, 0xC5, 0x63, 0x0E,
  0xB8, 0xC7, 0x1E, 0x2D, 0x6F, 0x09, 0xA4, 0x53,
  /* S-box 6 */
  0xC1, 0xAF, 0x92, 0x68, 0x0D, 0x34, 0xE7, 0x5B,
  0xAF, 0x42, 0x7C, 0x95, 0x61, 0xDE, 0x0B, 0x38,
  0x9E, 0xF5, 0x28, 0xC3, 0x70, 0x4A, 0x1D, 0xB6,
  0x43, 0x2C, 0x95, 0xFA, 0xBE, 0x17, 0x60, 0x8D,
  /* S-box 7 */
  0x4B, 0x2E, 0xF0, 0x8D, 0x3C, 0x97, 0x5A, 0x61,
  0xD0, 0xB7, 0x49, 0x1A, 0xE3, 0x5C, 0x2F, 0x86,
  0x14, 0xBD, 0xC3, 0x7E, 0xAF, 0x68, 0x05, 0x92,
  0x6B, 0xD8, 0x14, 0xA7, 0x95, 0x0F, 0xE2, 0x3C,
  /* S-box 8 */
  0xD2, 0x84, 0x6F, 0xB1, 0xA9, 0x3E, 0x50, 0xC7,
  0x1F, 0xD8, 0xA3, 0x74, 0xC5, 0x6B, 0x0E, 0x92,
  0x7B, 0x41, 0x9C, 0xE2, 0x06, 0xAD, 0xF3, 0x58,
  0x21, 0xE7, 0x4A, 0x8D, 0xFC, 0x90, 0x35, 0x6B
};

const uint8_t e_permtab[] ={
   4,  6,           /* 4 bytes in 6 bytes out*/
  32,  1,  2,  3,  4,  5,
   4,  5,  6,  7,  8,  9,
   8,  9, 10, 11, 12, 13,
  12, 13, 14, 15, 16, 17,
  16, 17, 18, 19, 20, 21,
  20, 21, 22, 23, 24, 25,
  24, 25, 26, 27, 28, 29,
  28, 29, 30, 31, 32,  1
};

const uint8_t p_permtab[] ={
   4,  4,              /* 32 bit -> 32 bit */
  16,  7, 20, 21,
  29, 12, 28, 17,
   1, 15, 23, 26,
   5, 18, 31, 10,
   2,  8, 24, 14,
  32, 27,  3,  9,
  19, 13, 30,  6,
  22, 11,  4, 25
};

const uint8_t ip_permtab[] ={
   8,  8,              /* 64 bit -> 64 bit */
  58, 50, 42, 34, 26, 18, 10, 2,
  60, 52, 44, 36, 28, 20, 12, 4,
  62, 54, 46, 38, 30, 22, 14, 6,
  64, 56, 48, 40, 32, 24, 16, 8,
  57, 49, 41, 33, 25, 17,  9, 1,
  59, 51, 43, 35, 27, 19, 11, 3,
  61, 53, 45, 37, 29, 21, 13, 5,
  63, 55, 47, 39, 31, 23, 15, 7
};

const uint8_t inv_ip_permtab[] ={
   8, 8,               /* 64 bit -> 64 bit */
  40, 8, 48, 16, 56, 24, 64, 32,
  39, 7, 47, 15, 55, 23, 63, 31,
  38, 6, 46, 14, 54, 22, 62, 30,
  37, 5, 45, 13, 53, 21, 61, 29,
  36, 4, 44, 12, 52, 20, 60, 28,
  35, 3, 43, 11, 51, 19, 59, 27,
  34, 2, 42, 10, 50, 18, 58, 26,
  33, 1, 41,  9, 49, 17, 57, 25
};

const uint8_t pc1_permtab[] ={
   8,  7,              /* 64 bit -> 56 bit*/
  57, 49, 41, 33, 25, 17,  9,
   1, 58, 50, 42, 34, 26, 18,
  10,  2, 59, 51, 43, 35, 27,
  19, 11,  3, 60, 52, 44, 36,
  63, 55, 47, 39, 31, 23, 15,
   7, 62, 54, 46, 38, 30, 22,
  14,  6, 61, 53, 45, 37, 29,
  21, 13,  5, 28, 20, 12,  4
};

const uint8_t pc2_permtab[] ={
   7,  6,           /* 56 bit -> 48 bit */
  14, 17, 11, 24,  1,  5,
   3, 28, 15,  6, 21, 10,
  23, 19, 12,  4, 26,  8,
  16,  7, 27, 20, 13,  2,
  41, 52, 31, 37, 47, 55,
  30, 40, 51, 45, 33, 48,
  44, 49, 39, 56, 34, 53,
  46, 42, 50, 36, 29, 32
};

const uint8_t splitin6bitword_permtab[] = {
   8,  8,           /* 64 bit -> 64 bit */
  64, 64,  1,  6,  2,  3,  4,  5,
  64, 64,  7, 12,  8,  9, 10, 11,
  64, 64, 13, 18, 14, 15, 16, 17,
  64, 64, 19, 24, 20, 21, 22, 23,
  64, 64, 25, 30, 26, 27, 28, 29,
  64, 64, 31, 36, 32, 33, 34, 35,
  64, 64, 37, 42, 38, 39, 40, 41,
  64, 64, 43, 48, 44, 45, 46, 47
};

const uint8_t shiftkey_permtab[] = {
   7,  7,           /* 56 bit -> 56 bit */
   2,  3,  4,  5,  6,  7,  8,  9,
  10, 11, 12, 13, 14, 15, 16, 17,
  18, 19, 20, 21, 22, 23, 24, 25,
  26, 27, 28,  1,
  30, 31, 32, 33, 34, 35, 36, 37,
  38, 39, 40, 41, 42, 43, 44, 45,
  46, 47, 48, 49, 50, 51, 52, 53,
  54, 55, 56, 29
```

```
};

#define ROTTABLE        0x7EFC
byte crypt[8];
byte plaintext [8];
byte keyword [] = { 0x5b,0x5a,0x57,0x67,0x6a,0x56,0x67,0x6e};

byte test[8];
int i,m;
int Dx =0;
uint32_t box=0,t;
String showprint = "";
uint8_t data[8];
/***************************************************************************


/*---------- Encoders ----------*/
#define c_LeftEncoderPinA 2
#define c_LeftEncoderPinB 40
volatile long _LeftEncoderTicks = 0L;

#define c_RightEncoderPinA 3
#define c_RightEncoderPinB 41
volatile long _RightEncoderTicks = 0L;
/*------------------------------*/

/*---- Motor Driver Setup ----*/
//Front right motor
int FRmotorA = 31;        //in1
int FRmotorB = 30;        //in2
int FRmotorSpeed = 5;   //ENABLE

//Back right motor
int BRmotorA = 47;        //in1
int BRmotorB = 46;        //in2
int BRmotorSpeed = 7;   //ENABLE

//Front left motor
int FLmotorA = 35;        //in1
int FLmotorB = 34;        //in2
int FLmotorSpeed = 4;   //ENABLE

//Back left motor
```

```
int BLmotorA = 49;        //in1
int BLmotorB = 48;        //in2
int BLmotorSpeed = 6;    //ENABLE
/*----------------------------------------*/

/*
 * ****************************************************
 * connectionInit() initialises the ESP8266 to station
 * mode then attempts to connect to the defined WiFi
 * network
 * ****************************************************
 */
boolean connectionInit(){
  boolean result = false;
  if(DEBUG){
    Serial.println("ESP8266 Initialisation Begin:");
  }

  //Library sets AT+CWMODE = 1 - Station mode
  if(DEBUG){
    if (wifi.setOprToStation()) {
        Serial.print("to station + softap ok\r\n");
    } else {
        Serial.print("to station + softap err\r\n");
    }
  }
  else{
    wifi.setOprToStation();
  }

  //Connect to WiFi network
  if(DEBUG){
    if (wifi.joinAP(SSID, PASSWORD)) {
        Serial.print("Join AP success\r\n");
        Serial.print("IP: ");
        Serial.println(wifi.getLocalIP().c_str());
    } else {
        Serial.print("Join AP failure\r\n");
    }
  }
  else{
    wifi.joinAP(SSID, PASSWORD);
  }
  //Set single connection mode to send initial data to webserver
  if(DEBUG){
    if (wifi.disableMUX()) {
        Serial.print("single ok\r\n");
    } else {
        Serial.print("single err\r\n");
    }
  }
  else{
    wifi.disableMUX();
  }
}

/*
 * ****************************************************
 * motorSetup() initialises the pins for the motor drivers
 * ****************************************************
 */
void motorSetup(){
  pinMode(FRmotorA,OUTPUT);
  pinMode(FRmotorB,OUTPUT);
  pinMode(FRmotorSpeed,OUTPUT);
  digitalWrite(FRmotorA,LOW);
  digitalWrite(FRmotorB,LOW);
  analogWrite(FRmotorSpeed,0);
  pinMode(FLmotorA,OUTPUT);
  pinMode(FLmotorB,OUTPUT);
  pinMode(FLmotorSpeed,OUTPUT);
  digitalWrite(FLmotorA,LOW);
  digitalWrite(FLmotorB,LOW);
  analogWrite(FLmotorSpeed,0);
  pinMode(BRmotorA,OUTPUT);
  pinMode(BRmotorB,OUTPUT);
  pinMode(BRmotorSpeed,OUTPUT);
  digitalWrite(BRmotorA,LOW);
  digitalWrite(BRmotorB,LOW);
  analogWrite(BRmotorSpeed,0);
  pinMode(BLmotorA,OUTPUT);
  pinMode(BLmotorB,OUTPUT);
  pinMode(BLmotorSpeed,OUTPUT);
  digitalWrite(BLmotorA,LOW);
  digitalWrite(BLmotorB,LOW);
  analogWrite(BLmotorSpeed,0);
}

/*
 * ****************************************************
 * Encoder Interrupt Service Routines
 * ****************************************************
 */
void leftMotorISR()
{
  if(digitalReadFast(c_LeftEncoderPinB)==1){
    _LeftEncoderTicks--;
  }
  else{
    _LeftEncoderTicks++;
  }
}

// Interrupt service routines for the right motor's quadrature
   encoder
void rightMotorISR()
{
  if(digitalReadFast(c_RightEncoderPinB)==1){
    _RightEncoderTicks++;
  }
  else{
    _RightEncoderTicks--;
  }
}

/*
 * ****************************************************
 * Setup function
 * ****************************************************
 */
void setup(void)
{
  Serial.begin(115200);
  if(DEBUG){
    Serial.print("setup begin\r\n");
  }
  delay(500);
  connectionInit();
  motorSetup();
  if(DEBUG){
    Serial.print("setup end\r\n");
  }
  pinMode(13,OUTPUT);
  digitalWrite(13,LOW);
  Dynamixel.begin(200000,9);  // Inicialize the servo at 1Mbps
     and Pin Control 2
  delay(1000);
  for(int i=1;i<8;i++){
    Dynamixel.setTempLimit(i,80);  // Set Max Temperature to 80
       Celcius
    Dynamixel.setVoltageLimit(i,65,160);  // Set Operating
       Voltage from 6.5v to 16v
    Dynamixel.setMaxTorque(i,800);        // 50% of Torque
    Dynamixel.setSRL(i,1);
  }

  // Motor quadrature encoders
  // Left encoder
  pinMode(c_LeftEncoderPinA, INPUT_PULLUP);       // sets pin A
     as input
  pinMode(c_LeftEncoderPinB, INPUT_PULLUP);       // sets pin B
     as input
  attachInterrupt(0, leftMotorISR, RISING);

  // Right encoder
  pinMode(c_RightEncoderPinA, INPUT_PULLUP);       // sets pin A
     as input
  pinMode(c_RightEncoderPinB, INPUT_PULLUP);       // sets pin B
     as input
  attachInterrupt(1, rightMotorISR, RISING);
}

/*
 * *******************************************************
 * Main Loop
 * *******************************************************
 */
void loop(void)
{
  unsigned long currentMillis = millis();
  if(((currentMillis - previousMillis) > interval) &&
     lastCmdCompleted) {
    previousMillis = currentMillis;
    sendGet();
    receiveData();    //Check if incoming data received from
       webserver
    limitAngles();
    moveRobotArm();
  }
  driveCheck();
}

/*
 * ****************************************************
 * moveRobotArm() sends commands to the AX-12 servos to move
 * to the commanded angle at a set speed
 * ****************************************************
 */
void moveRobotArm(){
  Dynamixel.moveSpeed(1,baseAngle,200);
  Dynamixel.moveSpeed(2,shoulderAngle,250);
  Dynamixel.moveSpeed(3,shoulderAngle,250);
  Dynamixel.moveSpeed(4,elbowAngle,250);
  Dynamixel.moveSpeed(5,elbowAngle,250);
  Dynamixel.moveSpeed(6,wristAngle,250);
  Dynamixel.moveSpeed(7,gripAngle,250);
}

/*
 * ****************************************************
 * limitAngles() prevents damage to robot by limiting
 * angles that the robotic arm joints can turn to
 * ****************************************************
 */
void limitAngles(){
  if(baseAngle<200){baseAngle=200;}
  if(baseAngle>800){baseAngle=800;}

  if(shoulderAngle<100){shoulderAngle=100;}
  if(shoulderAngle>700){shoulderAngle=700;}

  if(elbowAngle<500){elbowAngle=500;}
  if(elbowAngle>900){elbowAngle=900;}
  if(wristAngle<100){wristAngle=100;}
  if(wristAngle>950){wristAngle=950;}

  if(gripAngle<512){gripAngle=512;}
  if(gripAngle>700){gripAngle=700;}
}

/*
 * ***************************************************
 * driveCheck() uses the encoder counters to check if
 * the requested distance/angle has been moved
 * ***************************************************
 */
void driveCheck(){
  long rightCount = _RightEncoderTicks;
  long leftCount = _LeftEncoderTicks;

  if(fortyfiveFlag){
    if(directionChar == 'r'){
      if(rightCount<=R_Target){
        motorsOff('r');
      }
      if(leftCount>=L_Target){
        motorsOff('l');
      }
      if(rightCount<=R_Target && leftCount>=L_Target){
        fortyfiveFlag = false;
        lastCmdCompleted = true;
      }
    }
    else if(directionChar == 'l'){
      if(rightCount>=R_Target){
        motorsOff('r');
      }
      if(leftCount<=L_Target){
        motorsOff('l');
      }
      if(rightCount>=R_Target && leftCount<=L_Target){
        fortyfiveFlag = false;
        lastCmdCompleted = true;
      }
    }
  }

  if(moveFlag){
    if(directionChar == 'f'){
      if(rightCount>=Target){
        motorsOff('r');
      }
      if(leftCount>=Target){
        motorsOff('l');
      }
      if(rightCount>=Target && leftCount>=Target){
        moveFlag = false;
        lastCmdCompleted = true;
      }
    }
    else if(directionChar == 'b'){
      if(rightCount<=Target){
        motorsOff('r');
      }
      if(leftCount<=Target){
        motorsOff('l');
      }
      if(rightCount<=Target && leftCount<=Target){
        moveFlag = false;
        lastCmdCompleted = true;
      }
    }
  }
}

/*
 * ****************************************************
 * moveStraight() moves the robot forward or backwards in
 * a straight line for the commanded distance
 * ****************************************************
 */
void moveStraight(float dist){     //dist in mm
  _RightEncoderTicks = 0;
  _LeftEncoderTicks = 0;
  moveFlag = true;
  Target =  dist/0.6;
  if(dist<0){
    setDirection('b');
  }
  else{
    setDirection('f');
  }
  analogWrite(FLmotorSpeed,75);
  analogWrite(BLmotorSpeed,75);
  analogWrite(FRmotorSpeed,75);
  analogWrite(BRmotorSpeed,75);
}

/*
 * ****************************************************
 * motorsOff() turns off the left motors, right motors or
 * all motors as requested
 * ****************************************************
 */
void motorsOff(char x){
  if(x == 'l'){
    analogWrite(FLmotorSpeed,0);
    analogWrite(BLmotorSpeed,0);
  }
  else if(x == 'r'){
    analogWrite(FRmotorSpeed,0);
    analogWrite(BRmotorSpeed,0);
  }
  else if(x == 'a'){
    analogWrite(FLmotorSpeed,0);
    analogWrite(BLmotorSpeed,0);
    analogWrite(FRmotorSpeed,0);
    analogWrite(BRmotorSpeed,0);
  }
}

/*
 * ****************************************************
 * processCommand() takes the decrypted command string and
 * executes the commands
 * ****************************************************
 */
void processCommand(String command, String value){
  if(command == "LED"){
    LEDStatus = value.toInt();
    if(LEDStatus == 1){
      digitalWrite(13,HIGH);
    }
    else if(LEDStatus == 0){
      digitalWrite(13,LOW);
    }
    if(DEBUG){
      Serial.print("LED Command Received: ");
      Serial.println(value);
    }
  }
  else if(command == "speedbar"){        //distance value in mm
    speedValue = value.toInt();
    float temp = float(speedValue);
    if(speedValue == -1){
      motorsOff('a');
      moveFlag = false;
      fortyfiveFlag = false;
      lastCmdCompleted = true;
    }
    else if(speedValue == 0){ }
    else{
      moveStraight(temp);
    }
    if(DEBUG){
      Serial.print("Speed Command Received: ");
      Serial.println(value);
    }
  }
  else if(command == "turnbar"){
    turnValue = value.toInt();
    if(!moveFlag){
      if(turnValue<0){
        fortyFive('l',turnValue);
      }
      else if(turnValue>0){
        fortyFive('r',turnValue);
      }
    }
    if(DEBUG){
      Serial.print("Turn Command Received: ");
      Serial.println(value);
    }
  }
  else if(command == "power"){
    powerStatus = value.toInt();
    if(DEBUG){
      Serial.print("Power Command Received: ");
      Serial.println(value);
    }
  }
  else if(command == "servos"){
    String subStr = value.substring(0,3);
    baseAngle = subStr.toInt();
    baseAngle = map(baseAngle,0,300,0,1023);
    subStr = value.substring(3,6);
    shoulderAngle = subStr.toInt();
    shoulderAngle = map(shoulderAngle,0,300,0,1023);
    subStr = value.substring(6,9);
    elbowAngle = subStr.toInt();
    elbowAngle = map(elbowAngle,0,300,0,1023);
    subStr = value.substring(9,12);
    wristAngle = subStr.toInt();
    wristAngle = map(wristAngle,0,300,0,1023);
    subStr = value.substring(12);
    gripAngle = subStr.toInt();
    gripAngle = map(gripAngle,0,300,0,1023);
  }
  else{
    if(DEBUG){
      Serial.print("Command Unknown: ");
      Serial.println(command);
    }
  }
}

/*
 * ****************************************************
 * Function sendGet() sends http get request to webserver
 * ****************************************************
 */
void sendGet(){
  if(wifi.createTCP(HOST_NAME, HOST_PORT)){
     //Serial.print("create tcp ok\r\n");
  }
  else{
     //Serial.print("create tcp err\r\n");
  }

  String getCommand = "GET /arduino?";
  getCommand += " HTTP/1.1\r\nHost: ";
  getCommand += HOST_NAME;
  getCommand += "\r\nContent-Type: text/html; charset=utf-8\r\n\
     r\n";
  char cmdBuf[getCommand.length()+1];
  getCommand.toCharArray(cmdBuf,getCommand.length()+1);
  wifi.send((const uint8_t*)cmdBuf, strlen(cmdBuf));
}

/*
 * ****************************************************
 * receiveData() receives HTTP response from web server,
 * decrypts the data and then parses it into command data
 * ****************************************************
 */
void receiveData(){
  uint8_t buffer[1024] = {0};   //Create buffer for received
     data
  uint32_t len = wifi.recv(buffer, sizeof(buffer), 3000);  //
     Check for incoming data
  //copy received data
  for(int i=0;i<1024;i++){
    buffer2[i] = buffer[i];
  }

  //If data has been received then 'len' will be greater than 0
  if (len > 0) {
    Serial.println("Decrypting");
    for(int i = 0;i<8;i++){
      stringToHex(i);
      des_dec( plaintext, crypt, keyword);
      int cmdIndex = i*8;
      for(int j = 0;j<8;j++){
        cmdBuffer[cmdIndex+j] = plaintext[j];
      }
    }
    Serial.println("Done decrypting");
    if(DEBUG){
      Serial.print("Status:[");
      Serial.print(wifi.getIPStatus().c_str());
    Serial.println("]");

    Serial.print("Received Data :");
    Serial.print("[");
    for(uint32_t i = 0; i < len; i++) {
        Serial.print((char)buffer[i]);
    }
    Serial.print("]\r\n");
  }
  //Parse commands and values from decrypted response data
  char temp = 'a';
  int index = 1;
  temp = cmdBuffer[index];
  String dataIn;
  while(temp != ' '){
    String cmd = "";
    String val = "";
    while(temp != '='){
      cmd += temp;
      index++;
      temp = cmdBuffer[index];
    }
    index++;
    temp = cmdBuffer[index];
    while(temp != '&' && temp != ' '){
      val += temp;
      index++;
      temp = cmdBuffer[index];
    }
    //Save received data into string to send back for
       confirmation
    dataIn += cmd;
    dataIn += "=";
    dataIn += val;
    if(temp == '&'){
      index++;
      temp = cmdBuffer[index];
      dataIn += "&";
    }
    processCommand(cmd,val);     //Take action based on
       received command
  }
  return;
  }
}

/*
 * *****************************************************
 * fortyFive() executes a (45*num) degree turn
 * *****************************************************
 */
void fortyFive(char dir,int num){
  if(dir=='r'){      //r = turn right, l = turn left
    _LeftEncoderTicks = 0;
    _RightEncoderTicks = 0;
    L_Target = (100*num);
    R_Target = -(100*num);
    setDirection('r');
    fortyfiveFlag = true;
    analogWrite(FLmotorSpeed,55);
    analogWrite(BLmotorSpeed,55);
    analogWrite(FRmotorSpeed,55);
    analogWrite(BRmotorSpeed,55);
  }
  else if(dir=='l'){      //r = turn right, l = turn left
    _LeftEncoderTicks = 0;
    _RightEncoderTicks = 0;
    L_Target = -(100*num);
    R_Target = (100*num);
    setDirection('l');
    fortyfiveFlag = true;
    analogWrite(FLmotorSpeed,55);
    analogWrite(BLmotorSpeed,55);
    analogWrite(FRmotorSpeed,55);
    analogWrite(BRmotorSpeed,55);
  }
}

/*
 * *****************************************************
 * setDirection() sets the motor driver pins for desired
 * movement direction
 * *****************************************************
 */
void setDirection(char x){
  directionChar = x;
```

```
if(x=='f')                          // Move robot forward
{
  digitalWrite(FRmotorA, HIGH);
  digitalWrite(FRmotorB, LOW);
  digitalWrite(FLmotorA, HIGH);
  digitalWrite(FLmotorB, LOW);
  digitalWrite(BRmotorA, LOW);
  digitalWrite(BRmotorB, HIGH);
  digitalWrite(BLmotorA, LOW);
  digitalWrite(BLmotorB, HIGH);
}
else if(x=='b')                     // Move robot backward
{
  digitalWrite(FRmotorA, LOW);
  digitalWrite(FRmotorB, HIGH);
  digitalWrite(FLmotorA, LOW);
  digitalWrite(FLmotorB, HIGH);
  digitalWrite(BRmotorA, HIGH);
  digitalWrite(BRmotorB, LOW);
  digitalWrite(BLmotorA, HIGH);
  digitalWrite(BLmotorB, LOW);
}
else if(x=='l')                     // turn left
{
  digitalWrite(FRmotorA, HIGH);
  digitalWrite(FRmotorB, LOW);
  digitalWrite(FLmotorA, LOW);
  digitalWrite(FLmotorB, HIGH);
  digitalWrite(BRmotorA, LOW);
  digitalWrite(BRmotorB, HIGH);
  digitalWrite(BLmotorA, HIGH);
  digitalWrite(BLmotorB, LOW);
}
else if(x=='r')                     // turn right
{
  digitalWrite(FRmotorA, LOW);
  digitalWrite(FRmotorB, HIGH);
  digitalWrite(FLmotorA, HIGH);
  digitalWrite(FLmotorB, LOW);
  digitalWrite(BRmotorA, HIGH);
  digitalWrite(BRmotorB, LOW);
  digitalWrite(BLmotorA, LOW);
  digitalWrite(BLmotorB, HIGH);
  }
}

/*
 * ****************************************************
 * stringToHex() takes a buffer of individual characters
 * and combines each pair of characters into a hex value.
 * This is necessary due to the way the encrypted data is
 * formatted and transmitted by the web server
 * ****************************************************
 */
void stringToHex(int blockNumber){
  int index = 135+(blockNumber*16);
  int count = 0;
  for(int i = index;i<index+16;i+=2){
    char temp1 = buffer2[i];
    char temp2 = buffer2[i+1];
    uint16_t byte1 = 0;
    uint16_t byte2 = 0;
    uint32_t byteTotal = 0;

    if(temp1=='0'){byte1+=0;}
    else if (temp1=='1'){byte1+=1;}
    else if (temp1=='2'){byte1+=2;}
    else if (temp1=='3'){byte1+=3;}
    else if (temp1=='4'){byte1+=4;}
    else if (temp1=='5'){byte1+=5;}
    else if (temp1=='6'){byte1+=6;}
    else if (temp1=='7'){byte1+=7;}
    else if (temp1=='8'){byte1+=8;}
    else if (temp1=='9'){byte1+=9;}
    else if (temp1=='a'||temp1=='A'){byte1+=10;}
    else if (temp1=='b'||temp1=='B'){byte1+=11;}
    else if (temp1=='c'||temp1=='C'){byte1+=12;}
    else if (temp1=='d'||temp1=='D'){byte1+=13;}
    else if (temp1=='e'||temp1=='E'){byte1+=14;}
    else if (temp1=='f'||temp1=='F'){byte1+=15;}

    if(temp2=='0'){byte2+=0;}
    else if (temp2=='1'){byte2+=1;}
    else if (temp2=='2'){byte2+=2;}
    else if (temp2=='3'){byte2+=3;}
    else if (temp2=='4'){byte2+=4;}
    else if (temp2=='5'){byte2+=5;}
    else if (temp2=='6'){byte2+=6;}
    else if (temp2=='7'){byte2+=7;}
    else if (temp2=='8'){byte2+=8;}
    else if (temp2=='9'){byte2+=9;}
    else if (temp2=='a'||temp2=='A'){byte2+=10;}
    else if (temp2=='b'||temp2=='B'){byte2+=11;}
    else if (temp2=='c'||temp2=='C'){byte2+=12;}
    else if (temp2=='d'||temp2=='D'){byte2+=13;}
    else if (temp2=='e'||temp2=='E'){byte2+=14;}
    else if (temp2=='f'||temp2=='F'){byte2+=15;}
    byte2 = byte2;
    byteTotal = (byte1*16) + byte2;
    crypt[count] = byteTotal;
    count++;
  }
}

/**************************************************************************

/**************************************************************************

/**************************************************************************

/*
 * Remaining functions are for decryption algorithm as detailed
    at
 * http://arduino-projects4u.com/des-algorithm/
 */
/**************************************************************************

/**************************************************************************

/**************************************************************************


void permute(const uint8_t *ptable, const uint8_t *in, uint8_t *
   out){
  uint8_t ob; /* in-bytes and out-bytes */
  uint8_t byte, bit; /* counter for bit and byte */
  ob = ptable[1];
  ptable = &(ptable[2]);
  for(byte=0; byte<ob; ++byte){
    uint8_t x,t=0;
    for(bit=0; bit<8; ++bit){
      x=*ptable++ -1 ;
        t<<=1;
      if((in[x/8]) & (0x80>>(x%8)) ){
        t|=0x01;
      }
    }
    out[byte]=t;test[byte]=t;
  }
}

/*****************************************************************************

void changeendian32(uint32_t * a){
  *a = (*a & 0x000000FF) << 24 |
     (*a & 0x0000FF00) <<  8 |
     (*a & 0x00FF0000) >>  8 |
     (*a & 0xFF000000) >> 24;
box=((*a & 0x000000FF) << 24)|
     (*a & 0x0000FF00) <<  8 |
     (*a & 0x00FF0000) >>  8 |
     (*a & 0xFF000000) >> 24;
}

/*****************************************************************************

static inline
void shiftkey(uint8_t *key){
  uint8_t k[7];
  memcpy(k, key, 7);
  permute((uint8_t*)shiftkey_permtab, k, key);
        if (DEBUG == true) {
        Serial.print ("CD[");Serial.print(m);Serial.print ("]
           56 xits = ");
        for (int j=0;j<7;j++){
        if (test[j]<0x10) Serial.print("0");
        Serial.print(test[j],HEX);Serial.print(" ");
        print_binary(test[j],8);Serial.print(" ");
        }
        Serial.println();
        }
```

```
}

/*************************************************************************

/*************************************************************************

static inline
uint64_t splitin6bitwords(uint64_t a){
  uint64_t ret=0;
  a &= 0x0000ffffffffffffLL;
  permute((uint8_t*)splitin6bitword_permtab, (uint8_t*)&a, (
     uint8_t*)&ret);
  return ret;
}

/*************************************************************************


static inline
uint8_t substitute(uint8_t a, uint8_t * sbp){
  uint8_t x;
  x = sbp[a>>1];
  x = (a&1)?x&0x0F:x>>4;
  return x;

}

/*************************************************************************


uint32_t des_f(uint32_t r, uint8_t* kr){
  uint8_t i;
  uint32_t ret;
  uint64_t data;
  uint8_t *sbp; /* sboxpointer */
  permute((uint8_t*)e_permtab, (uint8_t*)&r, (uint8_t*)&data);
        showprint ="E       48 bits = ";printout1(0,6);
  for(i=0; i<7; ++i) {((uint8_t*)&data)[i] ^= kr[i];}
        if (DEBUG == true) {
        Serial.print ("ExorKS 48 bits = ");
        for (int j=0;j<6;j++){
        if (((uint8_t*)&data)[j]<0x10) Serial.print("0");
        Serial.print(((uint8_t*)&data)[j],HEX);Serial.print(" ")
            ;
        print_binary(((uint8_t*)&data)[j],8);Serial.print(" ");
        }
        Serial.println();
        }

  /* Sbox substitution */
  data = splitin6bitwords(data);
  sbp=(uint8_t*)sbox;
  for(i=0; i<8; ++i){
    uint8_t x;
    x = substitute(((uint8_t*)&data)[i], sbp);
    t<<=4;
    t |= x;
    sbp += 32;
  }
  changeendian32(&t);
        if (DEBUG == true) {
        Serial.print ("Sbox    32 bits = ");

        if (box/0x1000000<0x10) Serial.print("0");
        Serial.print(box/0x1000000,HEX);Serial.print(" ");
        print_binary(box/0x1000000,8);Serial.print(" ");

        if (box/0x10000&0xFF<0x10) Serial.print("0");
        Serial.print(box/0x10000&0xFF,HEX);Serial.print(" ");
        print_binary(box/0x10000&0xFF,8);Serial.print(" ");

        if (((box/0x100)&0xFF)<0x10) Serial.print("0");
        Serial.print(box/0x100&0xFF,HEX);Serial.print(" ");
        print_binary(box/0x100,8);Serial.print(" ");

        if (box&0xFF<0x10) Serial.print("0");
        Serial.print(box&0xFF,HEX);Serial.print(" ");
        print_binary(box&0xFF,8);
        Serial.println();
        }
  permute((uint8_t*)p_permtab,(uint8_t*)&t, (uint8_t*)&ret);
        showprint = "P        32 bits = "; printout1(0,4);
  return ret;
}
```

```
/***************************************************************************

void des_enc(void* out, const void* in, const void* key){
#define R *((uint32_t*)&(data[4]))
#define L *((uint32_t*)&(data[0]))
  uint8_t kr[6],k[7];
  permute((uint8_t*)ip_permtab, (uint8_t*)in, data);
        showprint = "L[0]    32 bits = "; printout1(0,4);
        showprint = "R[0]    32 bits = "; printout1(4,8);
  permute((uint8_t*)pc1_permtab, (const uint8_t*)key, k);
        showprint = "CD[0]   56 bits = "; printout1(0,7);
  for(i=0; i<8; i++){

                Dx=i*2+1;
                if (DEBUG == true) { Serial.print("Round ");
                   Serial.println(Dx); }
    shiftkey(k);
    if(ROTTABLE&((1<<((i<<1)+0))) ) shiftkey(k);
    permute((uint8_t*)pc2_permtab, k, kr);
                showprint = "KS      48 bits = "; printout1(0,6);
    L ^= des_f(R, kr);
                showprint = "L[i]    32 bits = "; printout2(0,4);
                showprint = "R[i]    32 bits = "; printout2(4,8);

                Dx=i*2+2;
                if (DEBUG == true) { Serial.print("Round ");
                   Serial.println(Dx); }
    shiftkey(k);
    if(ROTTABLE&((1<<((i<<1)+1))) ) shiftkey(k);
    permute((uint8_t*)pc2_permtab, k, kr);
                showprint = "KS      48 bits = "; printout1(0,6);
    R ^= des_f(L, kr);
                showprint = "L[i]    32 bits = "; printout2(0,4);
                showprint = "R[i]    32 bits = "; printout2(4,8);
  }
  /* L <-> R*/
  R ^= L;
  L ^= R;
  R ^= L;
        showprint = "LR[16]  64 bits = "; printout2(0,8);
  permute((uint8_t*)inv_ip_permtab, data, (uint8_t*)out);
        showprint = "Crypt   64 bits = "; printout1(0,8);
```

```
}

/*******************************************************************

void des_dec(void* out, const void* in, const uint8_t* key){
#define R *((uint32_t*)&(data[4]))
#define L *((uint32_t*)&(data[0]))
  uint8_t kr[6],k[7];
  permute((uint8_t*)ip_permtab, (uint8_t*)in, data);
        showprint = "L[0]   32 bits = "; printout1(0,4);
        showprint = "R[0]   32 bits = "; printout1(4,8);
  permute((uint8_t*)pc1_permtab, (const uint8_t*)key, k);
        showprint = "CD[0]  56 bits = "; printout1(0,7);
  for(i=7; i>=0; i--){

              Dx=i*2+2;
              if (DEBUG == true) { Serial.print("Round ");
                 Serial.println(Dx); }
              permute((uint8_t*)pc1_permtab, (const uint8_t*)
                 key, k);
              for (m=1;m<Dx+1;m++){
              shiftkey(k);
              if(ROTTABLE&(1<<(m-1))) shiftkey(k);
              }
    permute((uint8_t*)pc2_permtab, k, kr);
              showprint = "KS     48 bits = "; printout1(0,6);
    L ^= des_f(R, kr);
              showprint = "L[i]   32 bits = "; printout2(0,4);
              showprint = "R[i]   32 bits = "; printout2(4,8);

              Dx=i*2+1;
              if (DEBUG == true) { Serial.print("Round ");
                 Serial.println(Dx); }
              permute((uint8_t*)pc1_permtab, (const uint8_t*)
                 key, k);
              for (m=1;m<Dx+1;m++){
    shiftkey(k);
              if(ROTTABLE&(1<<(m-1))) shiftkey(k);
              }
    permute((uint8_t*)pc2_permtab, k, kr);
              showprint = "KS     48 bits = "; printout1(0,6);
    R ^= des_f(L, kr);
```

```
                showprint = "L[i]    32 bits = "; printout2(0,4);
                showprint = "R[i]    32 bits = "; printout2(4,8);
  }
  /* L <-> R*/
  R ^= L;
  L ^= R;
  R ^= L;
        showprint = "LR[16] 64 bits = "; printout2(0,8);
  permute((uint8_t*)inv_ip_permtab, data, (uint8_t*)out);
        showprint = "Plain  64 bits = "; printout1(0,8);
}
void print_binary(uint64_t v, int num_places)
{
    uint64_t mask=0, n;
    for (n=1; n<=num_places; n++)
    {
        mask = (mask << 1) | 0x00000001;
    }
    v = v & mask;  // truncate v to specified number of places
    while(num_places)
    {
        if (v & (0x00000001 << num_places-1))
        {
             Serial.print("1");
        }
        else
        {
             Serial.print("0");
        }
        --num_places;
        if(((num_places%8) == 0) && (num_places != 0))
        {
            Serial.print(" ");
        }

    }
}
void printout1(int min,int max) {
        if (DEBUG == true) {
        Serial.print (showprint);
        for (int j=min;j<max;j++){
        if (test[j]<0x10) Serial.print("0");
        Serial.print(test[j],HEX);Serial.print(" ");
        print_binary(test[j],8);Serial.print(" ");
        }
        Serial.println();
        }
}
void printout2(int min,int max) {
        if (DEBUG == true) {
        Serial.print (showprint);
        for (int j=min;j<max;j++){
        if (data[j]<0x10) Serial.print("0");
        Serial.print(data[j],HEX);Serial.print(" ");
        print_binary(data[j],8);Serial.print(" ");
        }
        Serial.println();
        }
}
```

Bibliography

[1] (2015) Esp8266 - at command reference. Room-15. [Online]. Available: https://room-15.github.io/blog/2015/03/26/esp8266-at-command-reference/

[2] (2017) Arduino products. Arduino. [Online]. Available: https://www.arduino.cc/en/Main/Products

[3] (2017) Iso/osi model in communication networks. studytonight. Studytonight.com. [Online]. Available: http://www.studytonight.com/computer-networks/complete-osi-model

[4] "Signaling." *Columbia Electronic Encyclopedia, 6th Edition*, p. 1, 2017.

[5] (2017) Sql commands and functions. Microsoft. [Online]. Available: https://msdn.microsoft.com/en-us/library/aa978483(v=vs.71).aspx

[6] (2017) What is cloud computing? a beginners guide. Microsoft. [Online]. Available: https://azure.microsoft.com/en-au/overview/what-is-cloud-computing/

[7] M. Ahemd, M. Shah, and A. Wahid, "Iot security: A layered approach for attacks defenses," in *2017 International Conference on Communication Technologies (ComTech)*, 2017.

[8] J. Bartje. (2016) The top 10 iot application areas - based on rea iot projects. [Online]. Available: https://iot-analytics.com/top-10-iot-project-application-areas-q3-2016/

[9] I. A. Board. (2015) Architectural considerations in smart object networking. [Online]. Available: https://tools.ietf.org/html/rfc7452

[10] "Industry trend analysis - iot key themes 2017 - enterprise drives focus on cost cutting," Business Montitor International, 2015.

[11] "Economic analysis - the internet of things: Slow burner, but game changer," Business Montitor International, 2017.

[12] A. Foster, "Messaging technologies for the industrial internet and the internet of things," PrismTech, 2013.

[13] E. J. Hurst, "Evolutions in telemedicine: From smoke signals to mobile health solutions," *Journal of Hospital Librarianship*, vol. 16, no. 2, pp. 174–185, April 2016.

[14] "Information technology - open systems interconnection - basic reference model: The basic model," International Organisation for Standardization, 1996.

[15] R. W. Lucky, "Wires and wireless," *iEEE Spectrum*, p. 32, November 2012.

[16] J. Manyika, M. Chui, P. Bisson, J. Woetzel, R. Dobbs, J. Bughin, and D. Aharon, "The internet of things: Mapping the value beyond the hype," McKinsey Global Institute, 2015.

[17] D. McFarlane. (2015) The origin of the internet of things. [Online]. Available: http://www.redbite.com/the-origin-of-the-internet-of-things/

[18] P. Mell and T. Grance, "The nist definition of cloud computing," National Institute of Standards and Technology, 2011.

[19] Robottini. (2011) Dynamixel ax-12a and arduino: how to use the serial port. [Online]. Available: http://robottini.altervista.org/dynamixel-ax-12a-and-arduino-how-to-use-the-serial-port

[20] K. Rose, S. Eldridge, and L. Chapin, "The internet of things: An overview," Internet Society, White Paper, 2015.

[21] S. Vashi, J. Ram, J. Modi, S. Verma, and C. Prakash, "Internet of things (iot): A vision, architectural elements, and security issues," in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2017.

[22] R. Weatherley. (2017) Arduino libraries. [Online]. Available: https://github.com/rweather/arduinolibs