**Chapter 1**

**1. Introduction to Wireless Sensor Networks**

**1.1 Identify the requirements for the real world problems.**

Wireless sensors are standard measurement tools equipped with transmitters to convert signals from process control instruments into a radio transmission. The radio signal is interpreted by a receiver which then converts the wireless signal to a specific, desired output, such as an analog current or data analysis via computer software.

**Safety**: Wireless instruments can be used in locations that are difficult to access due to extreme conditions such as high temperature, pH, pressure, etc. Using wireless sensors, operators can continuously supervise processes in hazardous environments and report the data back to an operator in a monitoring facility located at a safe distance away. Wireless measurement is also useful for obtaining data in hard to access locations.

**Convenience**: Wireless sensors can be used to form a web/network that would allow an engineer to monitor a number of different locations from one station. This provides a centralized control of a factory. Additionally, a number of wireless sensors have the ability to create a unique web page making up-to-the-minute data, accessible anywhere in the world.

**Reduce Costs**: Wireless process control can reduce the cost of monitoring and running a factory by eliminating the need for extension wire, conduit, and other costly accessories.

**Selecting a Wireless Sensor**

There are a number of items to consider when selecting a wireless measurement instrument.

**Type of Measurement:** It is important to understand what is being measured. Wireless transmitters (which incorporate wireless process measurement and control) typically have a unique function. Sensors are specifically designed for temperature, pressure, flow, etc., and must be selected accordingly.

**Accuracy and Response Time:** How accurate does the measurement need to be, and how quickly should the measurement be updated? Most wireless sensors are as accurate as their wired counterparts; however the readings are typically transmitted every few seconds to preserve battery power. If instantaneous measurement is necessary, this must be taken into consideration when selecting the wireless transmitter because certain models may not offer the desired response time.

**Range**: The range of wireless sensors varies widely. Some are designed for short-range, indoor applications of a few hundred feet, while other sensors can transmit data to a receiver located miles away. Regardless of the sensors capability, the range of a wireless signal is always limited by obstructions. Transmitting through machines, walls, and structures degrades signal strength and reduces range capability.

Frequency: The frequency of radio transmission is also important to consider. Laws vary by country and region as to which parts of the wireless spectrum are available for use without specific licenses. In the USA 915MHz, 2.4GHz (WiFi) are the major frequencies factories can use to transmit signals. As part of the industrial, scientific, and medical band, users do not need a radio license to operate on these frequencies. In Europe, wireless products typically operate on 868MHz or 2.4GHz. Due to regulatory requirements, products may only be available in certain regions.

**Sensor mote**

A sensor node, also known as a mote (chiefly in [North America](https://en.wikipedia.org/wiki/North_America)), is a node in a [sensor network](https://en.wikipedia.org/wiki/Sensor_network) that is capable of performing some processing, gathering sensory information and communicating with other connected nodes in the network. A mote is a node but a node is not always a mote.

**Components**: The main components of a sensor node are a [microcontroller](https://en.wikipedia.org/wiki/Microcontroller), [transceiver](https://en.wikipedia.org/wiki/Transceiver), external [memory](https://en.wikipedia.org/wiki/Memory), [power source](https://en.wikipedia.org/wiki/Electric_power) and one or more [sensors](https://en.wikipedia.org/wiki/Sensors).

**Controller**: The controller performs tasks, processes data and controls the functionality of other components in the sensor node. While the most common controller is a [microcontroller](https://en.wikipedia.org/wiki/Microcontroller), other alternatives that can be used as a controller are: a general purpose [desktop](https://en.wikipedia.org/wiki/Desktop_computer) [microprocessor](https://en.wikipedia.org/wiki/Microprocessor), [digital signal processors](https://en.wikipedia.org/wiki/Digital_signal_processors), [FPGAs](https://en.wikipedia.org/wiki/Field_Programmable_Gate_Array) and [ASICs](https://en.wikipedia.org/wiki/Application-specific_integrated_circuit). A microcontroller is often used in many [embedded systems](https://en.wikipedia.org/wiki/Embedded_systems) such as sensor nodes because of its low cost, flexibility to connect to other devices, ease of programming, and low power consumption. A general purpose microprocessor generally has a higher power consumption than a microcontroller, therefore it is often not considered a suitable choice for a sensor node. [1]

**Transceiver**: Sensor nodes often make use of [ISM band](https://en.wikipedia.org/wiki/ISM_band), which gives free [radio](https://en.wikipedia.org/wiki/Radio), spectrum allocation and global availability. The possible choices of wireless transmission media are [radio frequency](https://en.wikipedia.org/wiki/Radio_frequency) (RF), [optical communication](https://en.wikipedia.org/wiki/Optical_communication) (laser) and [infrared](https://en.wikipedia.org/wiki/Infrared). Lasers require less energy , but need [line-of-sight](https://en.wikipedia.org/wiki/Line-of-sight_propagation) for [communication](https://en.wikipedia.org/wiki/Communication) and are sensitive to atmospheric conditions. Infrared, like lasers, needs no [antenna](https://en.wikipedia.org/wiki/Antenna_(radio)) but it is limited in its [broadcasting](https://en.wikipedia.org/wiki/Broadcasting) capacity. Radio frequency-based communication is the most relevant that fits most of the WSN applications. WSNs tend to use license-free communication frequencies: 173, 433, 868, and 915 [MHz](https://en.wikipedia.org/wiki/MHz); and 2.4 [GHz](https://en.wikipedia.org/wiki/GHz). The functionality of both [transmitter](https://en.wikipedia.org/wiki/Transmitter) and [receiver](https://en.wikipedia.org/wiki/Receiver_(radio)) are combined into a single device known as a [transceiver](https://en.wikipedia.org/wiki/Transceiver). Transceivers often lack unique identifiers. The operational states are transmit, receive, idle, and sleep. Current generation transceivers have built-in [state machines](https://en.wikipedia.org/wiki/State_machines) that perform some operations automatically.

**External memory**: From an energy perspective, the most relevant kinds of memory are the on-chip memory of a microcontroller and [Flash memory](https://en.wikipedia.org/wiki/Flash_memory)—off-chip [RAM](https://en.wikipedia.org/wiki/RAM) is rarely, if ever, used. Flash memories are used due to their cost and storage capacity. Memory requirements are very much application dependent. Two categories of memory based on the purpose of storage are: user memory used for storing application related or personal data, and program memory used for programming the device. Program memory also contains identification data of the device if present.

**Power source**: A wireless sensor node is a popular solution when it is difficult or impossible to run a mains supply to the sensor node. However, since the wireless sensor node is often placed in a hard-to-reach location, changing the battery regularly can be costly and inconvenient. An important aspect in the development of a wireless sensor node is ensuring that there is always adequate energy available to power the system. The sensor node [consumes power](https://en.wikipedia.org/wiki/Power_consumption) for sensing, communicating and data processing. More energy is required for data communication than any other process.

**Sensors**: [Sensors](https://en.wikipedia.org/wiki/Sensor) are used by wireless sensor nodes to capture data from their environment. They are hardware devices that produce a measurable response to a change in a physical condition like temperature or pressure. Sensors measure physical data of the parameter to be monitored and have specific characteristics such as accuracy, sensitivity etc. The continual [analog signal](https://en.wikipedia.org/wiki/Analog_signal) produced by the sensors is digitized by an [analog-to-digital converter](https://en.wikipedia.org/wiki/Analog-to-digital_converter) and sent to controllers for further processing. Some sensors contain the necessary electronics to convert the raw signals into readings which can be retrieved via a digital link (e.g. I2C, SPI) and many convert to units such as °C. Most sensor nodes are small in size, consume little energy, operate in high volumetric densities, be autonomous and operate unattended, and be adaptive to the environment. As wireless sensor nodes are typically very small electronic devices, they can only be equipped with a limited power source of less than 0.5-2 ampere-hour and 1.2-3.7 volts. Sensors are classified into three categories: passive, omnidirectional sensors; passive, narrow-beam sensors; and active sensors. Passive sensors sense the data without actually manipulating the environment by active probing. They are self powered; that is, energy is needed only to amplify their analog signal. Active sensors actively probe the environment, for example, a sonar or radar sensor, and they require continuous energy from a power source. [2]

**1.2 Applications of Wireless Sensor Networks**

**Area monitoring:**

Area monitoring is a common application of WSNs. In area monitoring, the WSN is deployed over a region where some phenomenon is to be monitored. A military example is the use of sensors to detect enemy intrusion; a civilian example is the [geo-fencing](https://en.wikipedia.org/wiki/Geo-fence) of gas or oil pipelines. [2]

**Health care monitoring:**

There are several types of sensor networks for medical applications: implanted, wearable, and environment-embedded. Implantable medical devices are those that are inserted inside the human body. Wearable devices are used on the body surface of a human or just at close proximity of the user. Environment-embedded systems employ sensors contained in the environment. Possible applications include body position measurement, location of persons, overall monitoring of ill patients in hospitals and at home. Devices embedded in the environment track the physical state of a person for continuous health diagnosis, using as input the data from a network of depth cameras, a [sensing floor](https://en.wikipedia.org/wiki/Sensing_floor), or other similar devices. Body-area networks can collect information about an individual's health, fitness, and energy expenditure.[[3]](https://en.wikipedia.org/wiki/Wireless_sensor_network#cite_note-3)[[4]](https://en.wikipedia.org/wiki/Wireless_sensor_network#cite_note-4) In health care applications the privacy and authenticity of user data has prime importance. Especially due to the integration of sensor networks, with IoT, the user authentication becomes more challenging; however, a solution is presented in recent work.[2]

**Environmental/Earth sensing:**

There are many applications in monitoring environmental parameters, examples of which are given below. They share the extra challenges of harsh environments and reduced power supply.

**Air pollution monitoring:**

Wireless sensor networks have been deployed in several cities ([Stockholm](https://en.wikipedia.org/wiki/Stockholm), [London](https://en.wikipedia.org/wiki/London), and [Brisbane](https://en.wikipedia.org/wiki/Brisbane)) to monitor the concentration of [dangerous gases for citizens](https://en.wikipedia.org/wiki/Air_pollution). These can take advantage of the ad hoc wireless links rather than wired installations, which also make them more mobile for testing readings in different areas[3]

**Forest fire detection:**

A network of Sensor Nodes can be installed in a forest to detect when a [fire](https://en.wikipedia.org/wiki/Forest_fire) has started. The nodes can be equipped with sensors to measure temperature, humidity and gases which are produced by fire in the trees or vegetation. The early detection is crucial for a successful action of the firefighters; thanks to Wireless Sensor Networks, the fire brigade will be able to know when a fire is started and how it is spreading.

**Landslide detection:**

A [landslide](https://en.wikipedia.org/wiki/Landslide) detection system makes use of a wireless sensor network to detect the slight movements of soil and changes in various parameters that may occur before or during a landslide. Through the data gathered it may be possible to know the impending occurrence of landslides long before it actually happens.

**Water quality monitoring:**

[Water quality](https://en.wikipedia.org/wiki/Water_quality) monitoring involves analyzing water properties in dams, rivers, lakes and oceans, as well as underground water reserves. The use of many wireless distributed sensors enables the creation of a more accurate map of the water status, and allows the permanent deployment of monitoring stations in locations of difficult access, without the need of manual data retrieval.[[7]](https://en.wikipedia.org/wiki/Wireless_sensor_network#cite_note-7)

**Natural disaster prevention:**

Wireless sensor networks can be effective in preventing adverse consequences of [natural disasters](https://en.wikipedia.org/wiki/Natural_disaster), like floods. Wireless nodes have been deployed successfully in rivers, where changes in water levels must be monitored in real time.

**Machine health monitoring:**

Wireless sensor networks have been developed for machinery condition-based maintenance (CBM) as they offer significant cost savings and enable new functionality.Wireless sensors can be placed in locations difficult or impossible to reach with a wired system, such as rotating machinery and untethered vehicles. [3]

**Data center monitoring:**

Due to the high density of server racks in a data center, often cabling and IP addresses are an issue. To overcome that problem more and more racks are fitted out with wireless temperature sensors to monitor the intake and outtake temperatures of racks. As [ASHRAE](https://en.wikipedia.org/wiki/ASHRAE) recommends up to six temperature sensors per rack, meshed wireless temperature technology gives an advantage compared to traditional cabled sensors.

**Data logging:**

Wireless sensor networks also are used for the collection of data for monitoring of environmental information. This can be as simple as monitoring the temperature in a fridge or the level of water in overflow tanks in nuclear power plants. The statistical information can then be used to show how systems have been working. The advantage of WSNs over conventional loggers is the "live" data feed that is possible.

**Water/waste water monitoring:**

Monitoring the quality and level of water includes many activities such as checking the quality of underground or surface water and ensuring a country’s water infrastructure for the benefit of both human and animal. It may be used to protect the wastage of water.

Wireless sensor networks can be used to monitor the condition of civil infrastructure and related geo-physical processes close to real time, and over long periods through data logging, using appropriately interfaced sensors.

Wireless sensor networks are used to monitor wine production, both in the field and the cellar. [4]

**Chapter 2**

**2. Introduction to your Mini-Project Topic: Tank Tag (LO2)**

**2.1 Project Abstract:**

This program with arduino focuses on creating interactive gaming session .The rules of the game are pretty simple. Just shoot your laser at the opposing tank and hit it till the health points reach 0 or when the time runs out eventually. This consists of two tanks facing each other at every instance. The game is pretty simple and an excellent and a popular source of entertainment. Though the game can be conducted manually, it often needs elaborate preparations. The final output is envisioned to be a user-friendly interactive gaming session with which the user can get entertainment with value-addition. This tank tag game is based on the video game World Of Tanks. [7]

**2.2 Aims and Objectives:**

The main objective of this project is “Entertainment” as this project is a gameplay of two remote controlled robots having laser diodes to aim with servo motor and shoot at each other on the target’s LDR plate.[6]

**2.3 Scope of the project:**

In this gameplay the players try to knock each other out using their tanks. Each tank has 100 health points and each successful shot from the opponent on the receiver LDR plate reduces 10 health points. There is a time limit of 2 minutes set for the game to end and whoever has the most health points when the time has ended, wins the game. Or the other possibility is that one of the player gets knocked out even before the time is up, due to loss of all 100 health points.[8]

**2.4 Features of the project:**

**1. Real Time Data Exchange**

As the game progresses there is real time data exchange taking place to show the live health of both the users on a webpage and on their own gaming console (app) with the help of firebase realtime database.[9]

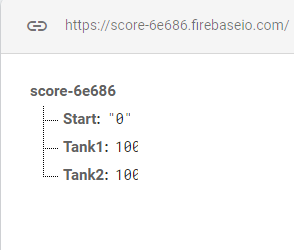


Figure 2.1: Tank Tag Realtime Database(Firebase).

**2. Tank Operates Wirelessly**

Instead of using a wired connection for the tank we have used a bluetooth module on each tank to avoid the difficulties caused due to the wires.[7]



Figure 2.2: Tank Tag Android Application.

**3. Score Board**

Live healths of both the tanks are displayed on a webpage.

Both the apps won’t display control buttons until the game is started from the website and after the game is over, to start a new game reset buttons is to be clicked from the web page and next game will start when the start button is clicked.[10]

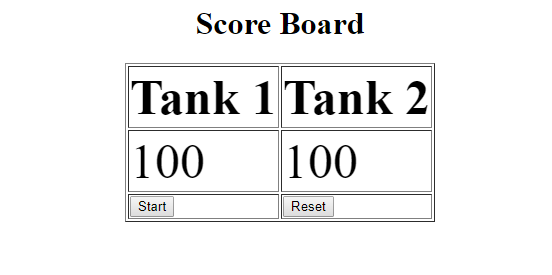


Figure 2.3: Tank Tag Scoreboard Web Page.

**Chapter 3**

**3. Review of Literature (LO2)**

A review of completed and ongoing research has been conducted to identify current knowledge or methodologies that may be appropriate for our project in various way.

Key sources that have been consulted in the literature review include:

Power supply. The Arduino processing boards may be powered from the USB port during project development. However it is highly recommended that an external power supply be employed. This will allow developing projects beyond the limited current capability of the USB port. Arduino www.arduino.cc recommends a power supply from 7–12 VDC with a 2.1 mm center positive plug.

Arduino software: You will also need the Arduino software called the Arduino Development Environment. It is available as a free download from the Arduino homepage.

The Arduino Uno R3 is a microcontroller board based on the ATmega328 ([datasheet](http://www.atmel.com/dyn/resources/prod_documents/doc8161.pdf)). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

Photo resistors, also known as light dependent resistors (LDR), are light sensitive devices most often used to indicate the presence or absence of light, or to measure the light intensity. In the dark, their resistance is very high, sometimes up to 1MΩ, but when the LDR sensor is exposed to light, the resistance drops dramatically, even down to a few [ohms](http://www.resistorguide.com/ohms-law/), depending on the light intensity

A Laser Diode is a semiconductor device similar to a light-emitting diode (LED). It uses p-n junction to emit coherent light in which all the waves are at the same frequency and phase. This coherent light is produced by the laser diode using a process termed as “Light Amplification by Stimulated Emission of Radiation”, which is abbreviated as LASER. And since a p-n junction is used to produce laser light, this device is named as a laser diode.

This dual bidirectional motor driver is based on the very popular L298 Dual H-Bridge Motor Driver IC. This module will allow you to easily and independently control two motors of up to 2A each in both directions.

HTML is required to build the Scoreboard (web page) in which we’ve used HTML tables and HTML buttons for controlling the Android application and displaying live scores during the game play.

Javascript is also used to fetch and send data to firebase when the database is updated or the two buttons “Start” and “Reset” are clicked.

App Inventor for Android is an open-source web application originally provided by Google, and now maintained by the Massachusetts Institute of Technology, which allows newcomers to computer programming to create software applications for the Android operating system, we used it to create controller for the tanks.

The Firebase Realtime Database is a cloud-hosted NoSQL database that lets you store and sync data between your users in real time.Firebase Realtime database is a cloud hosted database that supports multiple platforms Android, iOS and Web. All the data is stored in JSON.

**Chapter 4**

**4. System Description**

**4.1 Design**

**Block Diagram**

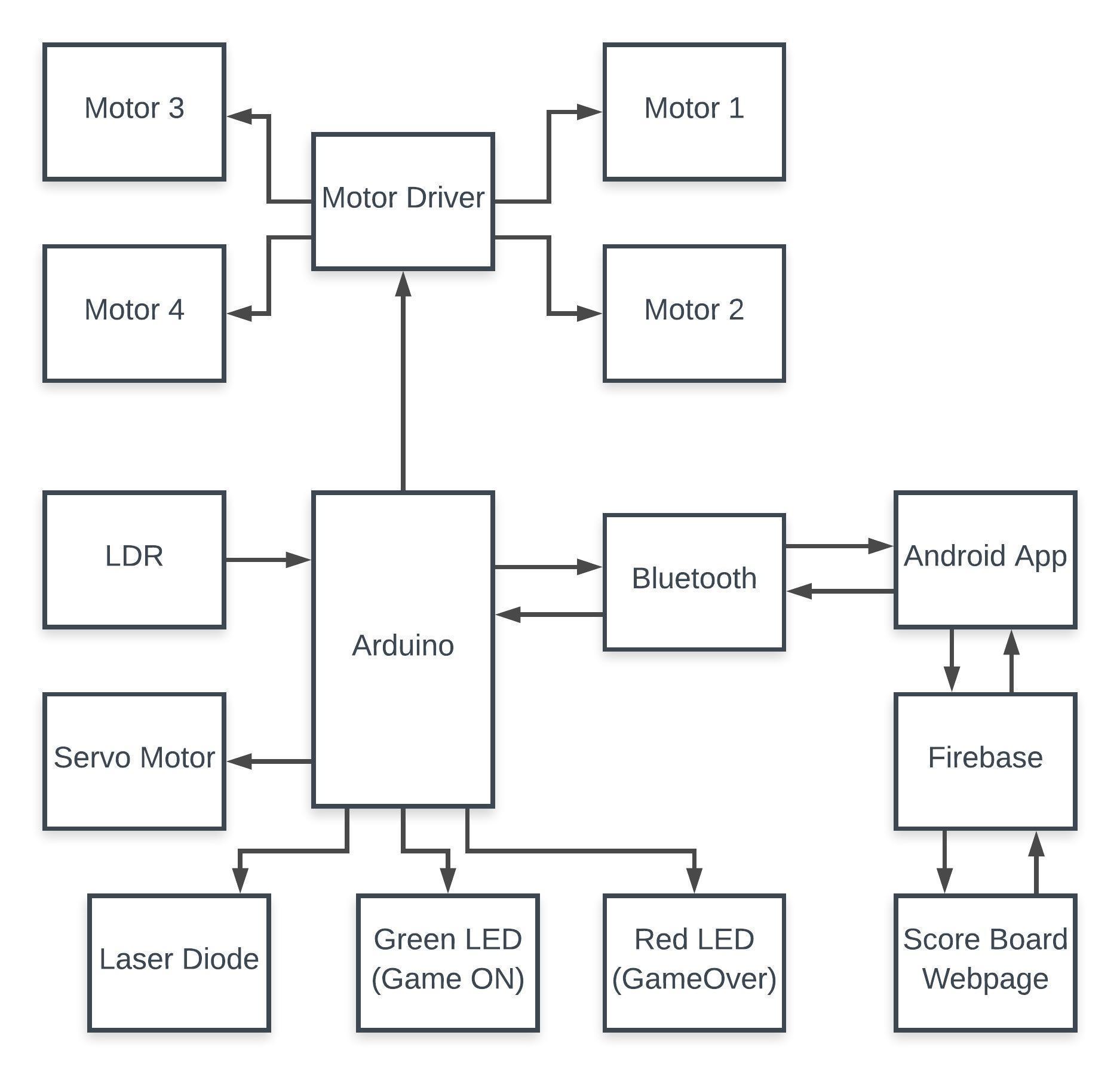


Figure 4.1: Block Diagram of Prototype (Tank Tag)

**4.2 Hardware, Software Requirements and Cloud Platforms used.**

**Hardware Requirements:**

**List of Hardware Requirements:**

Table 4.1 Hardware Requirements.

| **Sr No.** | **Component** | **Quantity** |
| --- | --- | --- |
| 1. | Arduino UNO | 2 |
| 2. | Bluetooth Module (HC-05) | 2 |
| 3. | DC Motors (300rpm) | 8 |
| 4. | LDR Sensors | 24 |
| 5. | Servo Motors (SG90) | 2 |
| 6. | Laser Diode | 2 |
| 7. | LED (Green and Red) | 2 (each) |
| 8. | Motor Driver(L289N) | 2 |
| 9. | AA battries | 16 |
| 10. | AA battery Case holder(4 battery holder) | 4 |
| 11. | Battery Connectors | 4 |

**1. Arduino Uno**

The Arduino Uno R3 is a microcontroller board based on the ATmega328 ([datasheet](http://www.atmel.com/dyn/resources/prod_documents/doc8161.pdf)). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.  
Revision 2 of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into [DFU mode](http://arduino.cc/en/Hacking/DFUProgramming8U2).



Figure 4.2: Arduino Uno

**2. HC-05 Bluetooth Module**

HC-05 has red LED which indicates connection status, whether the Bluetooth is connected or not. Before connecting to HC-05 module this red LED blinks continuously in a periodic manner. When it gets connected to any other Bluetooth device, its blinking slows down to two seconds.

We can connect 5V supply voltage as well since the module has on board 5 to 3.3 V regulator.

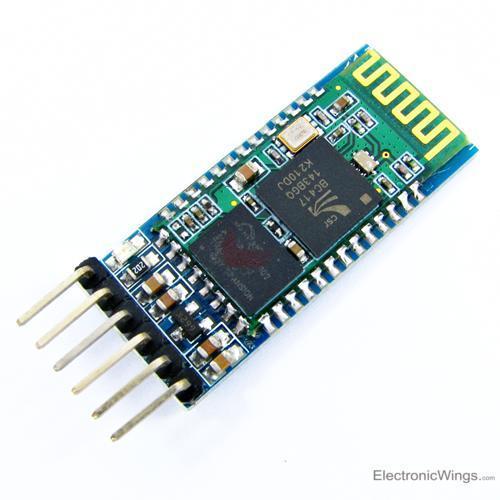


Figure 4.3: Bluetooth HC-05

**3. DC Motor 300 RPM**

300RPM 12V DC geared motors for robotics applications. Very easy to use and available in standard size. Nut and threads on shaft to easily connect and internal threaded shaft for easily connecting it to wheel.



Figure 4.4: DC Motor 300 RPM

**4. LDR Sensor**

Photo resistors, also known as light dependent resistors (LDR), are light sensitive devices most often used to indicate the presence or absence of light, or to measure the light intensity. In the dark, their resistance is very high, sometimes up to 1MΩ, but when the LDR sensor is exposed to light, the resistance drops dramatically, even down to a few [ohms](http://www.resistorguide.com/ohms-law/), depending on the light intensity. LDRs have a sensitivity that varies with the wavelength of the light applied and are nonlinear devices.

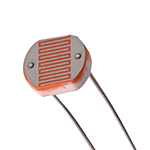


Figure 4.5: LDR Sensor

**5. Servo Motor**

A servo motor is an electrical device which can push or rotate an object with great precision. If you want to rotate and object at some specific angles or distance, then you use servo motor. It is just made up of simple motor which run through servo mechanism. If motor is used is DC powered then it is called DC servo motor, and if it is AC powered motor then it is called AC servo motor. We can get a very high torque servo motor in a small and light weight packages. Doe to these features they are being used in many applications like toy car, RC helicopters and planes, Robotics, Machine etc.



Figure 4.6: Servo Motor

**6. Laser Diode**

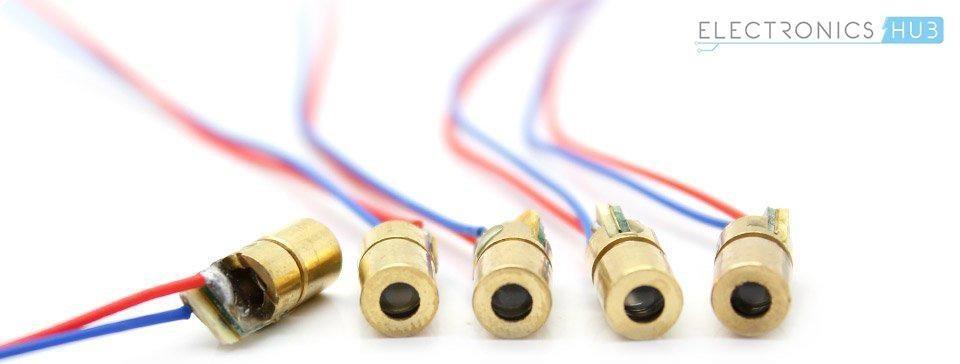
A Laser Diode is a semiconductor device similar to a light-emitting diode (LED). It uses p-n junction to emit coherent light in which all the waves are at the same frequency and phase. This coherent light is produced by the laser diode using a process termed as “Light Amplification by Stimulated Emission of Radiation”, which is abbreviated as LASER.

Figure 4.7: Laser Diode

**7. Led**

A light-emitting diode (LED) is a two-[lead](https://en.wikipedia.org/wiki/Lead_(electronics)) [semiconductor](https://en.wikipedia.org/wiki/Semiconductor) [light source](https://en.wikipedia.org/wiki/Light_source).When a suitable [current](https://en.wikipedia.org/wiki/Electric_current) is applied to the leads, [electrons](https://en.wikipedia.org/wiki/Electron) are able to recombine with [electron holes](https://en.wikipedia.org/wiki/Electron_hole) within the device, releasing energy in the form of [photons](https://en.wikipedia.org/wiki/Photon). This effect is called [electroluminescence](https://en.wikipedia.org/wiki/Electroluminescence), and the color of the light (corresponding to the energy of the photon) is determined by the energy [band gap](https://en.wikipedia.org/wiki/Band_gap) of the semiconductor.



Figure 4.8: LED

**8. L289N Motor Driver**

This dual bidirectional motor driver is based on the very popular L298 Dual H-Bridge Motor Driver IC. This module will allow you to easily and independently control two motors of up to 2A each in both directions.It is ideal for robotic applications and well suited for connection to a microcontroller requiring just a couple of control lines per m

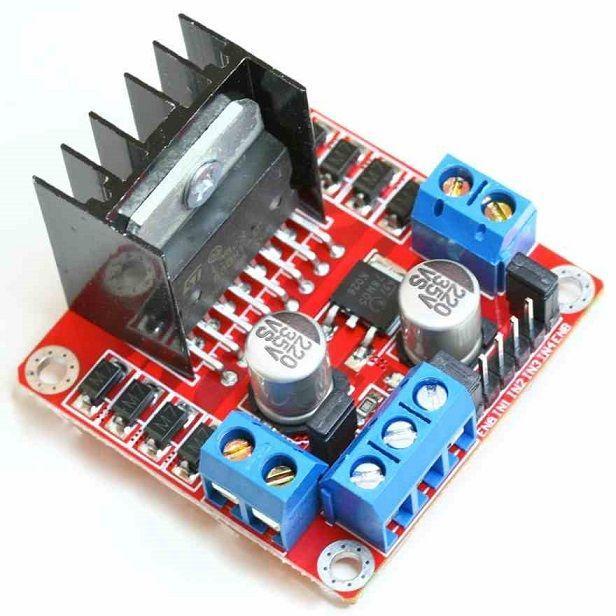


Figure 4.9: L289N Motor Driver

**9. AA Battery.**

AA batteries are common in portable [electronic devices](https://en.wikipedia.org/wiki/Electronics). An AA battery is composed of a single [electrochemical cell](https://en.wikipedia.org/wiki/Electrochemical_cell) that may be either a primary battery (disposable) or a rechargeable battery. The exact terminal [voltage](https://en.wikipedia.org/wiki/Voltage) and capacity of an AA size battery depend on cell chemistry; however, devices designed for AA will usually only take 1.5 V unless specified by the manufacturer.



Figure 4.10: AA Battery

**10. AA battery Case holder(4 battery holder).**

AA battery case holder are used for connecting AA batteries in series. here we’ve used 2x4AA battries to generate a 12v from 8xAA battries.

****

Figure 4.11: AA Battery case holder

**11. Battery connectors.**

Battery terminals are the electrical contacts used to connect a load or charger to a single cell or multiple-cell battery. These terminals have a wide variety of designs, sizes, and features that are often not well documented**.**



Figure 4.12: Battery Connector.

**Software Requirements**

**1. Arduino IDE:**

ARDUINO 1.8.7. The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board.

**2. MIT App inventor 2 Website**

App Inventor for Android is an open-source web application originally provided by Google, and now maintained by the Massachusetts Institute of Technology, which allows newcomers to computer programming to create software applications for the Android operating system.

**3. HTML**

Hypertext Markup Language is the standard markup language for creating web pages and web applications. With Cascading Style Sheets and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web.

**4. Firebase**

Firebase is a mobile and web app development platform that provides developers with a plethora of tools and services to help them develop high-quality apps, grow their user base, and earn more profit.Firebase is a mobile and web application development platform developed by Firebase, Inc. in 2011, then acquired by Google in 2014. As of October 2018, the Firebase platform has 18 products, which are used by 1.5 million apps.

**5. Web Browser**

To display the Scoreboard made with the help of HTML and Javascript.

**4.3 Implementation Methodology**

1. At first, we searched and studied all the possible sensors required for project.

2. After that we started implementing the prototype step by step starting beginning with the 2 tanks.

3. Later we started testing the tanks with the bluetooth module and android appliacation we created with the help of mit app inventor website.

4. Then after successfully implementing , we started testing the wireless connectivity between the app and the two tanks.

5. Then we tested our prototype connections for many possible cases of errors.

6. Once we finished testing the connections and working successfully, we tested the gameplay thoroughly and synchronised the tank healths with real time database(firebase).

7. Finally we covered the tank with cardboards and adjusted the laser diodes in line of sight of the opponents LDR sensor.

**4.4 Hardware Circuit Diagram**

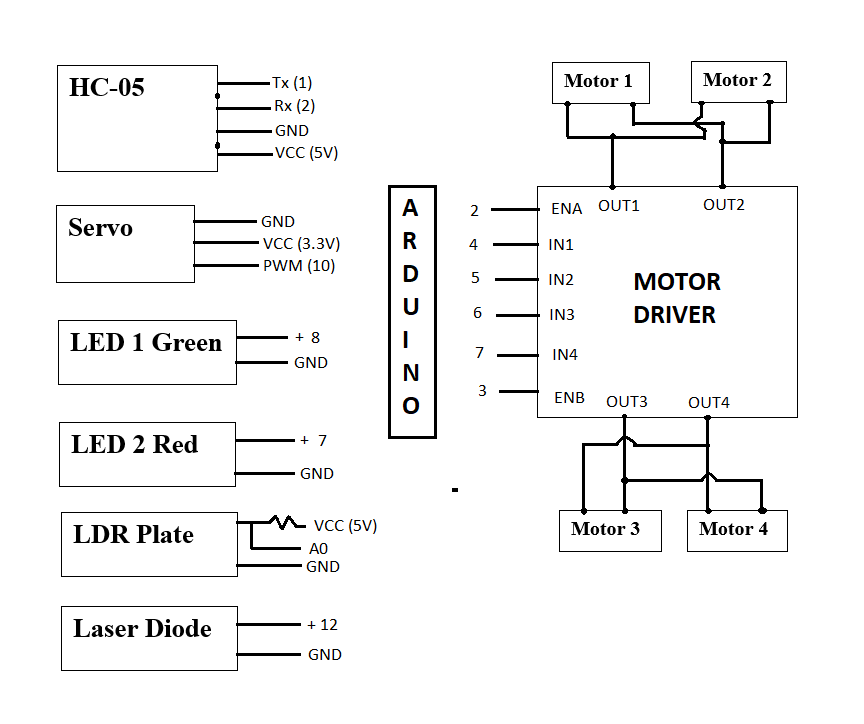


Figure 4.13: Circuit Diagram

**4.5 Code**

1. **Arduino Code.**

#include <Servo.h>

Servo myservo;

int lm1 = 4;

int lm2 = 5;

int rm1 = 6;

int rm2 = 7;

int fire = 12;

int fl = 3;

int bl = 2;

int pos=90;

int health=100;

int sensorPin = A0;

int gameon=8;

int gameoff=9;

int sensorValue = 0;

void setup()

{

//initlize the mode of the pins

pinMode(lm1,OUTPUT);

pinMode(lm2,OUTPUT);

pinMode(rm1,OUTPUT);

pinMode(rm2,OUTPUT);

pinMode(fire,OUTPUT);

pinMode(fl,OUTPUT);

pinMode(bl,OUTPUT);

pinMode(sensorPin,INPUT);

pinMode(gameoff,OUTPUT);

pinMode(gameon,OUTPUT);

myservo.attach(10);

myservo.write(pos);

//set the serial communication rate

Serial.begin(9600);

}

void loop()

{

//Motor PINS need this To ENABLE MOTION

analogWrite(fl,255);

analogWrite(bl,255);

if(health>0){

digitalWrite(gameon,HIGH);

digitalWrite(gameoff,LOW);

}

if(health <=0 ){

digitalWrite(gameoff,HIGH);

digitalWrite(gameon,LOW);

}

if(Serial.available()){

char val = Serial.read() ;//reads the signal from phone via bluetooth

/\*\*\*\*\*\*\*\*\*For Forward motion\*\*\*\*\*\*\*\*\*/

if(val == 'W')

{

digitalWrite(lm1,HIGH); digitalWrite(rm1,HIGH);

digitalWrite(lm2,LOW); digitalWrite(rm2,LOW);

}

if(val == 'w'){

digitalWrite(lm1,LOW); digitalWrite(rm1,LOW);

digitalWrite(lm2,LOW); digitalWrite(rm2,LOW);

}

/\*\*\*\*\*\*\*\*\*For Backward Motion\*\*\*\*\*\*\*\*\*/

if(val == 'S')

{

digitalWrite(lm2,HIGH); digitalWrite(rm2,HIGH);

digitalWrite(lm1,LOW); digitalWrite(rm1,LOW);

}

if(val == 's'){

digitalWrite(lm1,LOW); digitalWrite(rm1,LOW);

digitalWrite(lm2,LOW); digitalWrite(rm2,LOW);

}

/\*\*\*\*\*\*\*\*\*RIGHT\*\*\*\*\*\*\*\*\*/

if(val == 'D')

{

digitalWrite(lm1,HIGH); digitalWrite(rm2,HIGH);

digitalWrite(lm2,LOW); digitalWrite(rm1,LOW);

}

if(val == 'd'){

digitalWrite(lm1,LOW); digitalWrite(rm1,LOW);

digitalWrite(lm2,LOW); digitalWrite(rm2,LOW);

}

/\*\*\*\*\*\*\*\*\*LEFT\*\*\*\*\*\*\*\*\*/

if(val == 'A')

{

Serial.println("Right");

digitalWrite(lm2,HIGH); digitalWrite(rm1,HIGH);

digitalWrite(lm1,LOW); digitalWrite(rm2,LOW);

}

if(val == 'a'){

digitalWrite(lm1,LOW); digitalWrite(rm1,LOW);

digitalWrite(lm2,LOW); digitalWrite(rm2,LOW);

}

/\*\*\*\*\*\*\*\*\*Fire\*\*\*\*\*\*\*\*/

if(val == 'F')

{ digitalWrite(fire,HIGH);

}

if(val == 'f'){

digitalWrite(fire,LOW);

}

if(val == 'B')

{

digitalWrite(lm1,LOW); digitalWrite(rm1,LOW);

digitalWrite(lm2,LOW); digitalWrite(rm2,LOW); }

if(val == 'b'){

digitalWrite(lm1,LOW); digitalWrite(rm1,LOW);

digitalWrite(lm2,LOW); digitalWrite(rm2,LOW);;

}

if(val == 'Q'){

pos+=10;

myservo.write(pos);

delay(20);

if(pos >= 180 ){

pos=180;

} }

if(val == 'E'){

pos-=10;

myservo.write(pos);

delay(20);

if(pos <= 0 ){

pos=0;

} } }

///////////ALWAYS CHECK READINGS FIRST OF THE ENVIRONMENT AND CHECK LASER READINGS TOO

sensorValue = analogRead(sensorPin);

if ((sensorValue >=140 && sensorValue <= 280)){

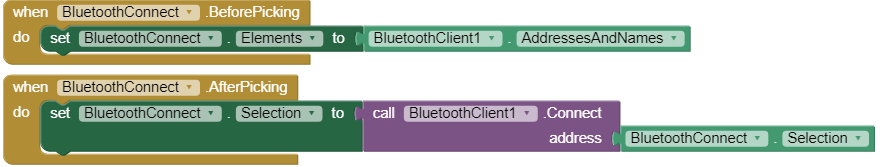
health-=10;

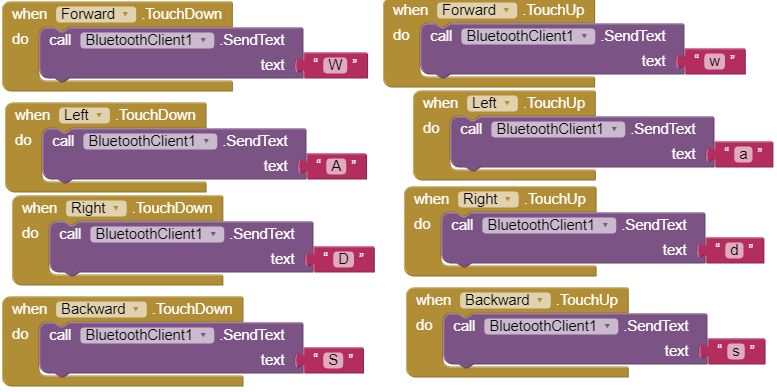
Serial.print("H");

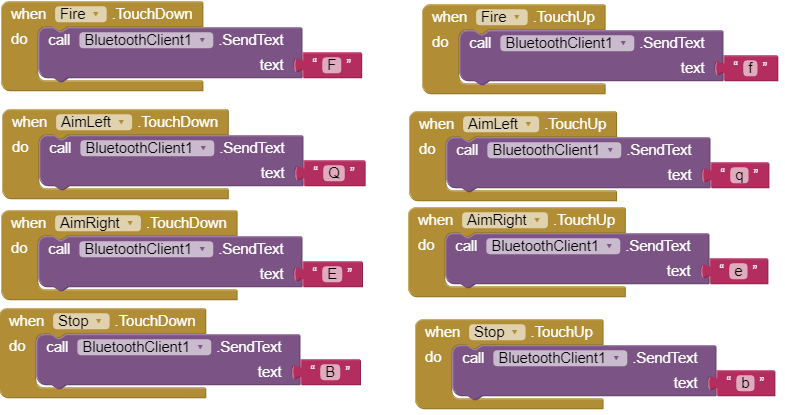
delay(1000);

}}

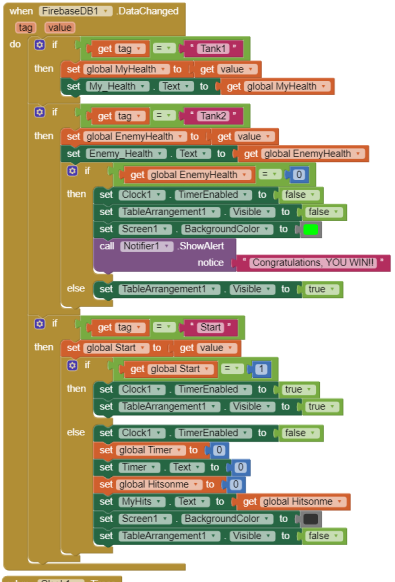
1. **App Inventor Blocks.**

****

****

****

****

****

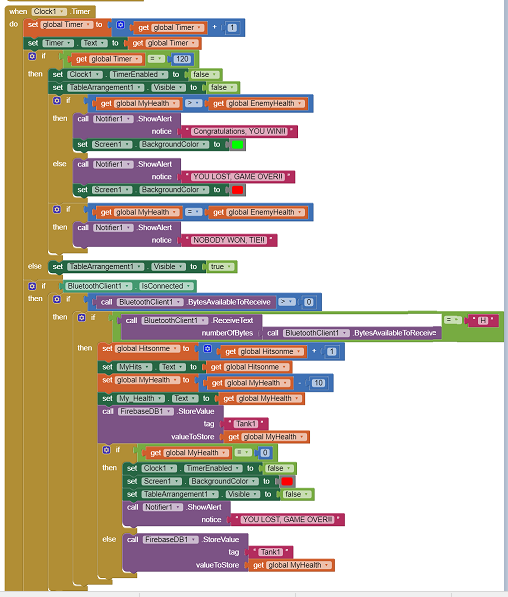
****

Figure 4.14: Android App building blocks.

1. **Web Page Code.**

**1. HTML Code**

<html>

<head>

<title>Tank Tag</title>

</head>

<body>

<script src="https://www.gstatic.com/firebasejs/5.9.2/firebase.js"></script>

<script>

var config = {

apiKey: "AIzaSyD\_-hC2nDSTTI9q6Oi9SsKeniDc2ygXRpk",

authDomain: "score-6e686.firebaseapp.com",

databaseURL: "https://score-6e686.firebaseio.com",

projectId: "score-6e686",

storageBucket: "score-6e686.appspot.com",

messagingSenderId: "1095007607745"

};

firebase.initializeApp(config);

</script>

<script src=index.js></script>

<h1 align="center"> Score Board </h1>

<table border=1 align="center" style="font-size: 50">

<tr>

<th>Tank 1</th>

<th>Tank 2</th>

</tr>

<tr>

<td id="Tank1health">100</td>

<td id="Tank2health">100</td>

</tr>

<td><button id="start" align="center" onclick="start()">Start</button></td>

<td > <button id="reset" align="center" onclick="reset()">Reset</button></td>

</table>

</body>

</html>

**2. Java Script**

var text = 100;

var button = document.getElementById("reset");

var T1h = document.getElementById("Tank1health");

var T2h = document.getElementById("Tank2health");

var t1 = fref = firebase.database().ref().child("Tank1");

t1.on('value',function(tank1snap){

Tank1health.innerText = tank1snap.val();

})

var t2 = fref = firebase.database().ref().child("Tank2");

t2.on('value',function(tank2snap){

Tank2health.innerText = tank2snap.val();

}

function reset(){

var val = text;

var fref = firebase.database().ref();

fref.child("Tank1").set(val);

fref.child("Tank2").set(val);

fref.child("Start").set("0");

window.alert("Database is Reset");

}

function start(){

var fref = firebase.database().ref();

fref.child("Start").set("1");

}

**4.6 Final Prototype**

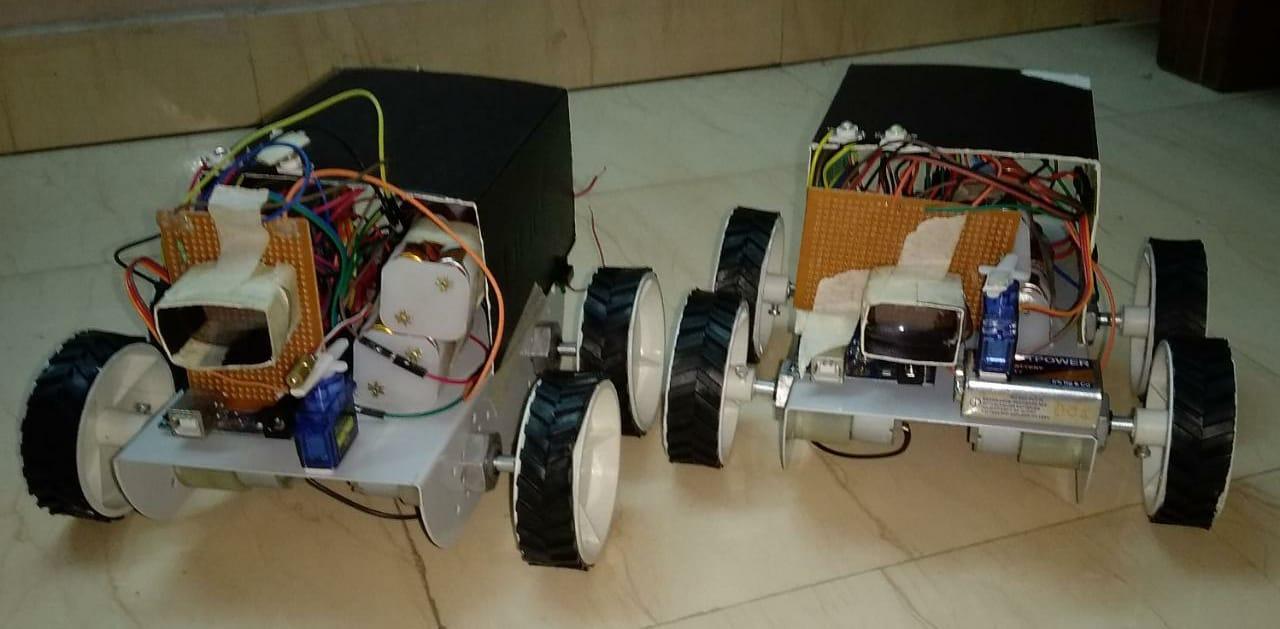
****

Figure 4.15: Final Prototype

**4.7 Future scope & Conclusion of the project:**

**Future Scope:**

This project can be further improved by providing a proper 3D printed body case for the tanks so that it will look much better than the current version.

More LDR sensors can be used in order to make the target hit from from all the 4 sides.

More number of servo motors can be used to give Vertical motion to the aiming gun of the tank.

Remote control android apps and Scoreboard can be designed more creatively for better look and feel.

To increase the range of controlling tanks remotely ESP8266-01(WiFi Module) can be used.

iOS application can be developed to control tanks and play the Tank Tag game.

**Conclusion**:

This project is completely based for entertainment purpose and can help refresh your mind. Project can be used for deploying real remote controlled military tanks in military applications, this way precious lives of the soldiers can be saved.

**4.8 Constraints for real time deployment**

1. In this project we have used only 12 LDR sensors on each tank but for this game to be more involving and interesting there should be much wider receiver on each tank.
2. iOS application is not developed to control tanks and play the Tank Tag game.
3. Remote control android apps and Scoreboard is not designed more creatively for better look and feel.
4. More number of servo motors are not used to give Vertical motion to the aiming gun of the tank.

**References:**

1. https://www.quora.com/What-are-some-real-world-problems-that-could-be-solved-with-programming-coding, accessed on February 15, 2019 2:40 pm.
2. https://en.wikipedia.org/wiki/Wireless\_sensor\_network, accessed on February 18, 2019 1:40 pm.
3. https://ieeexplore.ieee.org/document/8116858/, accessed on February 25, 2019 12:45 pm.
4. https://www.gettingsmart.com/2018/03/7-real-world-projects-that-allow-students-to-tackle-big-problems/ , accessed on February 28, 2019 4:15 pm.
5. How to control DC motor with L298N driver and Arduino. | Circuit Magic, accessed on March 8, 2019 7:20 pm.
6. https://create.arduino.cc/projecthub/gatoninja236/ww2-tank-laser-tag-sherman-panther-cdc98f?use\_route=project, accessed on March 12, 2019 3:00 pm.
7. <https://en.wikipedia.org/wiki/World_of_Tanks>
8. https://www.hackster.io/Yogeshmodi/bluetooth-controlled-car-e8c90e, accessed on March 18, 2019 7:30 am.
9. https://youtu.be/1dz0Aodmrz0, accessed on March 25, 2019 12:40 pm.
10. https://youtu.be/fU8QhoUJ\_sE, accessed on March 25, 2019 12:40 pm.
11. https://youtu.be/noB98K6A0TY, accessed on March 25, 2019 12:40 pm.

**Appendix**

**Assignments**

**A1: Assignment 1 (LO1, LO2, LO3, LO4, LO5)**

**Study of WSN opensource simulator- ContikiCooja**

**Contiki**

Contiki is an [operating system](https://en.wikipedia.org/wiki/Operating_system) for networked, memory-constrained systems with a focus on low-power wireless [Internet of Things](https://en.wikipedia.org/wiki/Internet_of_Things) devices. Extant uses for Contiki include systems for street lighting, sound monitoring for smart cities, radiation monitoring, and alarms.[[1]](https://en.wikipedia.org/wiki/Contiki#cite_note-primary-1) It is [open-source software](https://en.wikipedia.org/wiki/Open-source_software) released under a [BSD license](https://en.wikipedia.org/wiki/BSD_licenses).

Contiki was created by [Adam Dunkels](https://en.wikipedia.org/wiki/Adam_Dunkels) in 2002and has been further developed by a worldwide team of developers from Texas Instruments, Atmel, Cisco, [ENEA](https://en.wikipedia.org/wiki/ENEA_(Italy)), [ETH Zurich](https://en.wikipedia.org/wiki/ETH_Zurich), Redwire, [RWTH Aachen University](https://en.wikipedia.org/wiki/RWTH_Aachen_University), Oxford University, SAP, Sensinode, [Swedish Institute of Computer Science](https://en.wikipedia.org/wiki/Swedish_Institute_of_Computer_Science), ST Microelectronics, Zolertia, and many others. Contiki gained popularity because of its built in TCP/IP stack and lightweight preemptive scheduling over event-driven kernel which is a very motivating feature for IoT. The name Contiki comes from [Thor Heyerdahl](https://en.wikipedia.org/wiki/Thor_Heyerdahl)'s famous [Kon-Tiki](https://en.wikipedia.org/wiki/Kon-Tiki) raft.

Contiki provides multitasking and a built-in [Internet Protocol Suite](https://en.wikipedia.org/wiki/Internet_Protocol_Suite) (TCP/IP stack), yet needs only about 10 [kilobytes](https://en.wikipedia.org/wiki/Kilobyte) of [random-access memory](https://en.wikipedia.org/wiki/Random-access_memory) (RAM) and 30 kilobytes of [read-only memory](https://en.wikipedia.org/wiki/Read-only_memory) (ROM). A full system, including a [graphical user interface](https://en.wikipedia.org/wiki/Graphical_user_interface), needs about 30 kilobytes of RAM.

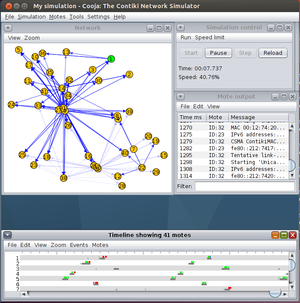


Figure A1.1: Contiki Cooja

**Hardware**:

Contiki is designed to run on types of hardware devices that are severely constrained in [memory](https://en.wikipedia.org/wiki/Memory), [power](https://en.wikipedia.org/wiki/Electric_power), processing power, and communication [bandwidth](https://en.wikipedia.org/wiki/Bandwidth_(computing)). A typical Contiki system has memory on the order of kilobytes, a power budget on the order of milliwatts, processing speed measured in megaHertz, and communication bandwidth on the order of hundreds of kilobits/second. Such systems include many types of [embedded systems](https://en.wikipedia.org/wiki/Embedded_system), and old [8-bit](https://en.wikipedia.org/wiki/8-bit) computers.

**Networking**:

Contiki provides three network mechanisms: the [uIP](https://en.wikipedia.org/wiki/UIP_(micro_IP)) TCP/IP stack, which provides [IPv4](https://en.wikipedia.org/wiki/IPv4) networking, the uIPv6 stack, which provides [IPv6](https://en.wikipedia.org/wiki/IPv6) networking, and the Rime stack, which is a set of custom lightweight networking protocols designed for low-power wireless networks. The IPv6 stack was contributed by Cisco and was, when released, the smallest IPv6 stack to receive the IPv6 Ready certification. The IPv6 stack also contains the Routing Protocol for Low power and Lossy Networks (RPL) routing protocol for low-power lossy IPv6 networks and the [6LoWPAN](https://en.wikipedia.org/wiki/6LoWPAN) header compression and adaptation layer for [IEEE 802.15.4](https://en.wikipedia.org/wiki/IEEE_802.15.4) links.

Rime is an alternative network stack, for use when the overhead of the IPv4 or IPv6 stacks is prohibitive. The Rime stack provides a set of communication primitives for low-power wireless systems. The default primitives are single-hop unicast, single-hop broadcast, multi-hop unicast, network flooding, and address-free data collection. The primitives can be used on their own or combined to form more complex protocols and mechanisms.

**Low Power Operation:**

Many Contiki systems are severely power-constrained. Battery operated wireless sensors may need to provide years of unattended operation and with little means to recharge or replace batteries. Contiki provides a set of mechanisms to reduce the power consumption of systems on which it runs. The default mechanism for attaining low-power operation of the radio is called ContikiMAC. With ContikiMAC, nodes can be running in low-power mode and still be able to receive and relay radio messages.

**Simulation:**

The Contiki system includes a network simulator called Cooja, which simulates networks of Contiki nodes. The nodes may belong to either of three classes: emulated nodes, where the entire hardware of each node is emulated, Cooja nodes, where the Contiki code for the node is compiled for and executed on the simulation host, or Java nodes, where the behavior of the node must be reimplemented as a Java class. One Cooja simulation may contain a mix of nodes from any of the three classes. Emulated nodes can also be used to include non-Contiki nodes in a simulated network.

**Programming Model:**

To run efficiently on small-memory systems, the Contiki programming model is based on [protothreads](https://en.wikipedia.org/wiki/Protothreads). A protothread is a memory-efficient programming abstraction that shares features of both [multithreading](https://en.wikipedia.org/wiki/Multithreading_(software)) and [event-driven programming](https://en.wikipedia.org/wiki/Event-driven_programming) to attain a low memory overhead of each protothread. The kernel invokes the protothread of a process in response to an internal or external event. Examples of internal events are timers that fire or messages being posted from other processes. Examples of external events are sensors that trigger or incoming packets from a radio neighbor.

Protothreads are cooperatively scheduled. Thus, a Contiki process must always explicitly yield control back to the kernel at regular intervals. Contiki processes may use a special protothread construct to block waiting for events while yielding control to the kernel between each event invocation.

**Features:**

Contiki supports per-process optional preemptive multithreading, [inter-process communication](https://en.wikipedia.org/wiki/Inter-process_communication) using message passing through events, as well as an optional [graphical user interface](https://en.wikipedia.org/wiki/Graphical_user_interface) (GUI) subsystem with either direct graphic support for locally connected terminals or networked virtual display with [Virtual Network Computing](https://en.wikipedia.org/wiki/Virtual_Network_Computing) (VNC) or over Telnet.

A full installation of Contiki includes the following features:

* Multitasking kernel
* Optional per-application [preemptive multithreading](https://en.wikipedia.org/wiki/Preemption_(computing))
* [Protothreads](https://en.wikipedia.org/wiki/Protothreads)
* [Internet Protocol Suite](https://en.wikipedia.org/wiki/Internet_Protocol_Suite) (TCP/IP) [networking](https://en.wikipedia.org/wiki/Computer_network), including [IPv6](https://en.wikipedia.org/wiki/IPv6)
* [Windowing system](https://en.wikipedia.org/wiki/Windowing_system) and GUI
* Networked remote display using [Virtual Network Computing](https://en.wikipedia.org/wiki/Virtual_Network_Computing)
* A [web browser](https://en.wikipedia.org/wiki/Web_browser) (claimed to be the world's smallest)
* Personal [web server](https://en.wikipedia.org/wiki/Web_server)
* Simple [telnet](https://en.wikipedia.org/wiki/Telnet) client
* [Screensaver](https://en.wikipedia.org/wiki/Screensaver)
* Contiki is supported by popular [SSL/TLS](https://en.wikipedia.org/wiki/Transport_Layer_Security) libraries such as [wolfSSL](https://en.wikipedia.org/wiki/WolfSSL), which includes a port in its 3.15.5 release.

**Ports:**

The Contiki operating system is ported to the following systems:

**Microcontrollers**

* [Atmel](https://en.wikipedia.org/wiki/Atmel) – [ARM](https://en.wikipedia.org/wiki/ARM_architecture), [AVR](https://en.wikipedia.org/wiki/Atmel_AVR)
* [NXP Semiconductors](https://en.wikipedia.org/wiki/NXP_Semiconductors) – LPC1768, LPC2103, [MC13224](http://www.nxp.com/products/microcontrollers-and-processors/arm-processors/kinetis-cortex-m/k-series/k3x-slcd-mcus/2.4-ghz-802.15.4-rf-and-32-bit-arm7-mcu-with-128kb-flash-96kb-ram:MC13224V)
* [Microchip](https://en.wikipedia.org/wiki/Microchip_Technology) – [dsPIC, PIC32](https://en.wikipedia.org/wiki/PIC_microcontroller) (PIC32MX795F512L)[Texas Instruments](https://en.wikipedia.org/wiki/Texas_Instruments) – [MSP430](https://en.wikipedia.org/wiki/TI_MSP430), CC2430, [CC2538](http://www.ti.com/tool/CC2538DK), [CC2630, CC2650](http://www.ti.com/tool/CC2650DK)

### [STMicroelectronics](https://en.wikipedia.org/wiki/STMicroelectronics) – [STM32 W](https://en.wikipedia.org/wiki/STM32#STM32_W)

### **Computers**

### [Apple](https://en.wikipedia.org/wiki/Apple_Inc.) – [II series](https://en.wikipedia.org/wiki/Apple_II_series)

### [Atari](https://en.wikipedia.org/wiki/Atari) – [8-bit](https://en.wikipedia.org/wiki/Atari_8-bit_family), [ST](https://en.wikipedia.org/wiki/Atari_ST), [Portfolio](https://en.wikipedia.org/wiki/Atari_Portfolio)

### [Casio](https://en.wikipedia.org/wiki/Casio) – [Pocket Viewer](https://en.wikipedia.org/wiki/Pocket_Viewer)

### [Commodore](https://en.wikipedia.org/wiki/Commodore_International) – [PET](https://en.wikipedia.org/wiki/Commodore_PET), [VIC-20](https://en.wikipedia.org/wiki/Commodore_VIC-20), [64](https://en.wikipedia.org/wiki/Commodore_64), [128](https://en.wikipedia.org/wiki/Commodore_128)

### [Tangerine Computer Systems](https://en.wikipedia.org/wiki/Tangerine_Computer_Systems) – [Oric](https://en.wikipedia.org/wiki/Oric)

### [NEC](https://en.wikipedia.org/wiki/NEC) – [PC-6001](https://en.wikipedia.org/wiki/NEC_PC-6001)

### [Sharp](https://en.wikipedia.org/wiki/Sharp_Corporation) – [Wizard](https://en.wikipedia.org/wiki/Sharp_Wizard)

### [Intel](https://en.wikipedia.org/wiki/Intel), [AMD](https://en.wikipedia.org/wiki/Advanced_Micro_Devices), [VIA](https://en.wikipedia.org/wiki/VIA_Technologies), many others – [x86](https://en.wikipedia.org/wiki/X86)-based [Unix-like](https://en.wikipedia.org/wiki/Unix-like) systems, atop [GTK+](https://en.wikipedia.org/wiki/GTK%2B), or more directly using an [X Window System](https://en.wikipedia.org/wiki/X_Window_System)

### **Game consoles**

### Atari – [Jaguar](https://en.wikipedia.org/wiki/Atari_Jaguar)

### [Game Park](https://en.wikipedia.org/wiki/Game_Park) – [GP32](https://en.wikipedia.org/wiki/GP32)

### [Nintendo](https://en.wikipedia.org/wiki/Nintendo) – [Game Boy](https://en.wikipedia.org/wiki/Game_Boy), [Game Boy Advance](https://en.wikipedia.org/wiki/Game_Boy_Advance), [Entertainment System](https://en.wikipedia.org/wiki/Nintendo_Entertainment_System) (NES)

### NEC – [TurboGrafx-16](https://en.wikipedia.org/wiki/TurboGrafx-16) Entertainment SuperSystem (PC Engine)

**A2: Assignment 2: 2 (LO1, LO2, LO3, LO4, LO5)**

**Study of WSN opensource simulator- Cupcarbon**

**CupCarbon**:

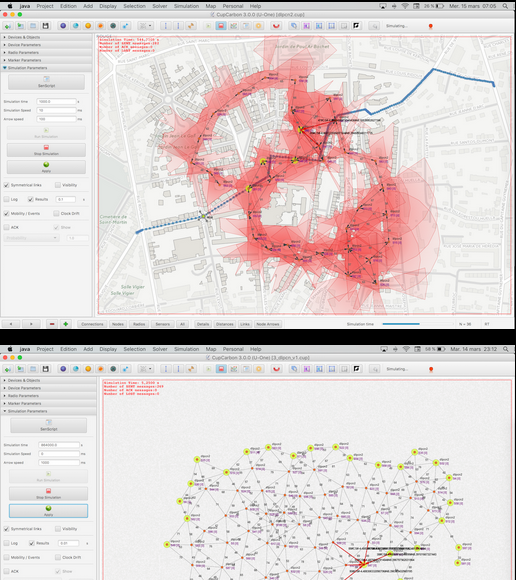
CupCarbon is a Smart City and Internet of Things Wireless Sensor Network (SCI-WSN) simulator. Its objective is to design, visualize, debug and validate distributed algorithms for monitoring, environmental data collection, etc., and to create environmental scenarios such as fires, gas, mobiles, and generally within educational and scientific projects. Not only it can help to visually explain the basic concepts of sensor networks and how they work; it may also support scientists to test their wireless topologies, protocols, etc. CupCarbon offers two simulation environments. The first simulation environment enables the design of mobility scenarios and the generation of natural events such as fires and gas as well as the simulation of mobiles such as vehicles and flying objects (e.g. UAVs, insects, etc.). The second simulation environment represents a discrete event simulation of wireless sensor networks which takes into account the scenario designed on the basis of the first environment. Networks can be designed and prototyped by an ergonomic and easy to use interface using the OpenStreetMap (OSM) framework to deploy sensors directly on the map. It includes a script called SenScript, which allows to program and to configure each sensor node individually. From this script, it is also possible to generate codes for hardware platforms such as Arduino/XBee.

Figure A2.1: CupCarbon

Steps to create a project:

Go to the directory having cupcarbon.jar file and open cmd.

Execute the code “java –jar cupcarbon.jar” and the application will open.

Click on Project and select New Project.

We can add sensors and we can write the code in SenScript.

In the device and objects tab, select the sensor and then in Device Parameters tab, select the script file and add it to the sensor.

To execute the file, start the simulation.

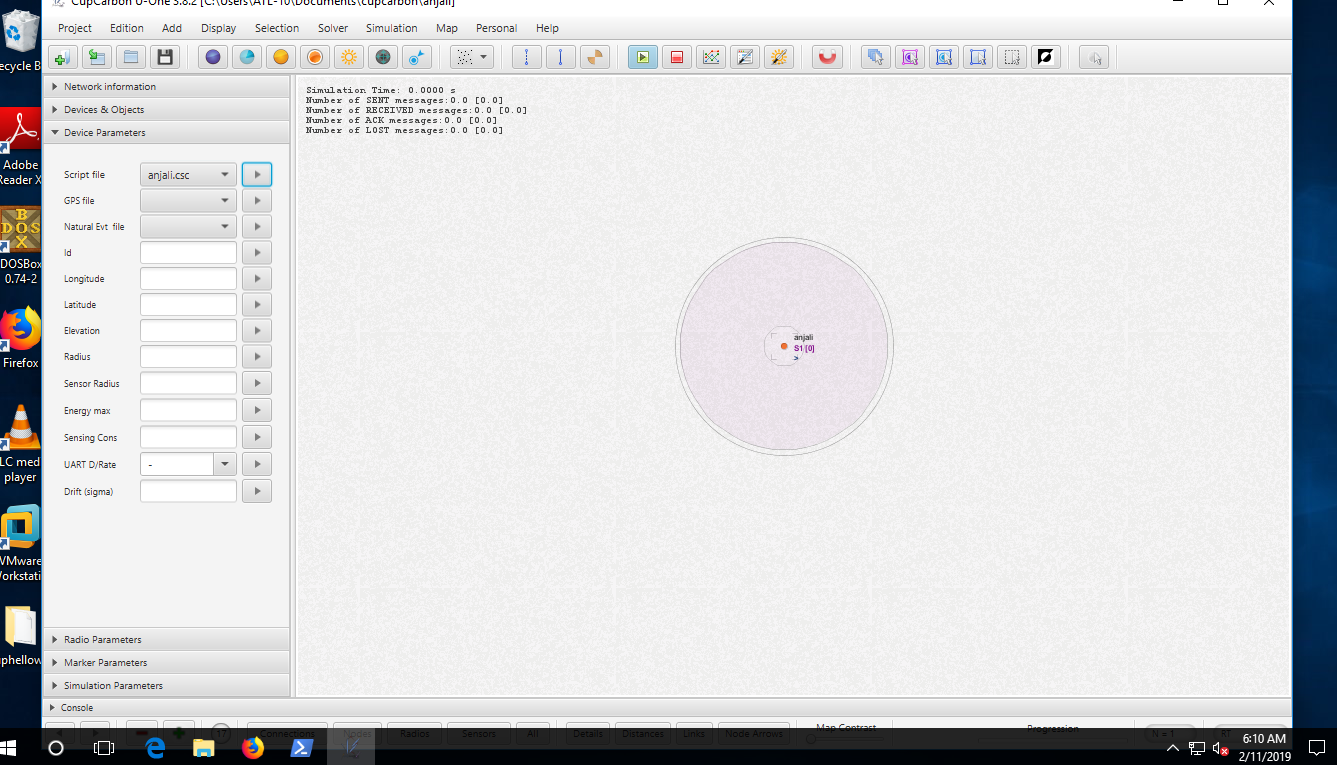
**Examples**:

CODE**:** loop

println Hello World

stop

OUTPUT:



CODE: loop

set a 7

for i 1 11

mult b $a $i

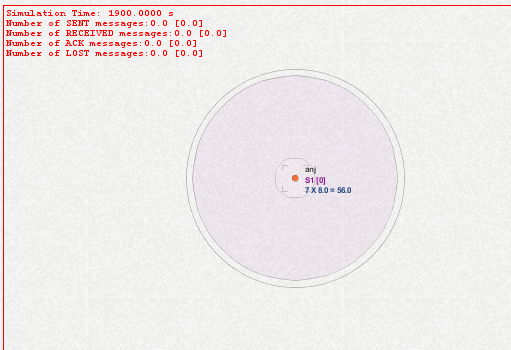
println $a \* $i = $b

delay 1000

end

stop

OUTPUT:



CODE: loop

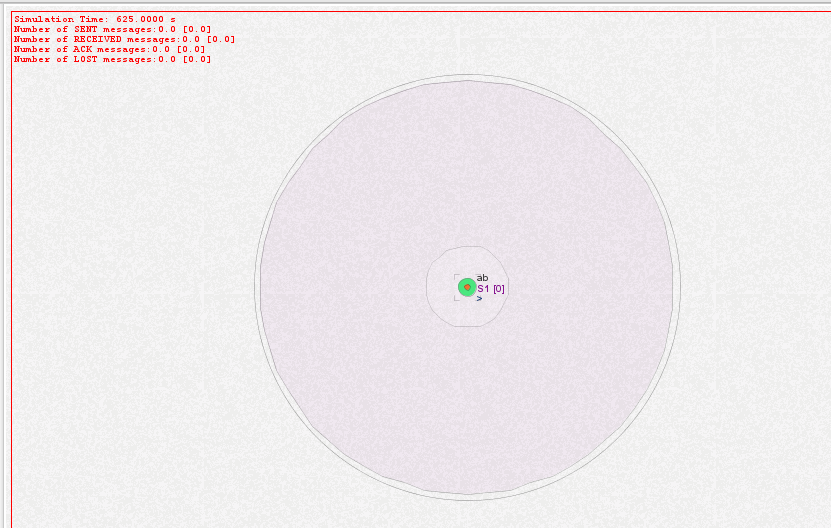
for i 0 15

led 13 $i

delay 5000

end

OUTPUT:



CODE: loop

rand x

if($x < 0.5)

mark 1

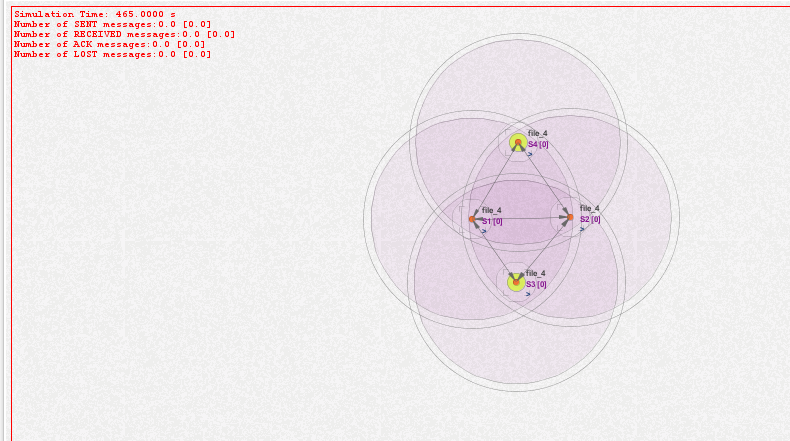
else

mark 0

end

delay 5000

OUTPUT:



CODE: // Transmitter

loop

send 1

delay 1000

send 0

delay 1000

// Receiver

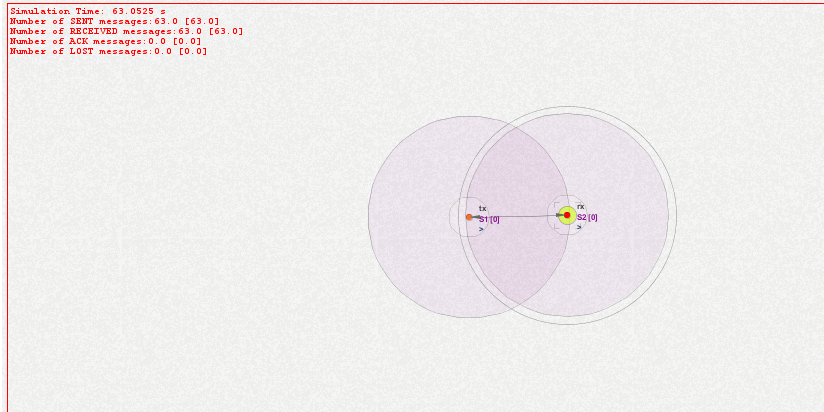
loop

wait

read v

mark $v

OUTPUT:



CODE: // Transmitter

loop

send 1

delay 1000

send 0

delay 1000

// Router 1

loop

wait

read v

send $v 3

// Router 2

loop

wait

read v

send $v 4

// Receiver

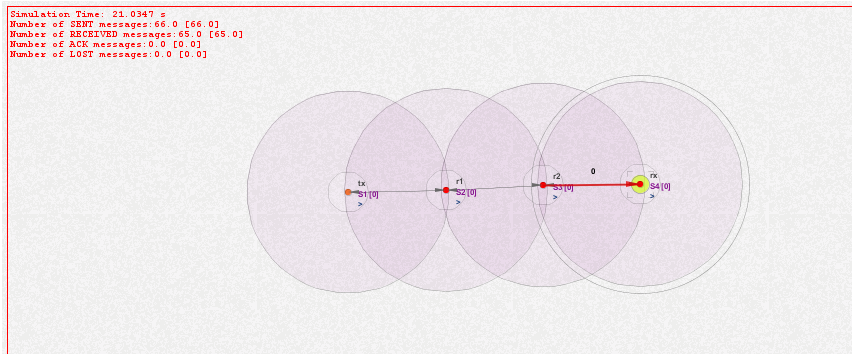
loop

wait

read v

mark $v

OUTPUT:



CODE:

// Transmitter

loop

send 1

delay 1000

send 0

delay 1000

// Receiver

loop

wait

read v

mark $v

OUTPUT: 