MANCHESTER 1824
The University of Manchester

Project Title: Robocone: Autonomous Deployment

Third Year Individual Project – Final Report

April 2019

Your Name: Ajaykumar Mudaliar

Your Identification Number: 9900559

Supervisor: Peter N Green

ROBOCONE: Robotic Traffic Cone

Swarming algorithms – Rule-based Autonomous Deployment

Abstract

Robocone is a system of a group of robotic traffic cones which will replace the traditional traffic cones used on site today. This project focuses on development of an autonomous deployment system for the Robocone system, meaning the robot-nodes should be capable of navigating the pre-mapped environment though measuring and sensing other robot-nodes and obstacle and changing formation accordingly. In this report, it is assumed that 16 robot nodes are going to be used and the transformation into different formations is done accordingly. Using MATLAB as a simulation tool by plotting the micro-movements in every transformation, the most efficient and fastest way of transformation is selected. Multiple MONAs robots represent the traffic cones (robot-nodes) in this project, they co-ordinate with a server for a master- slave system. Based on the previously conducted research on this project, a number of tests have been done to support the implementation of the project. Via the Arduino board and MONA robots it is instructed to perform movement and communicate between robot-nodes. To encourage swarm behaviour, sensors and radio modules are used to communicate with other nodes and position each node locally. A range of test is conducted to find the best solution for the sensing of other robots and obstacles. The robots are coded in Arduino such that they are able to perform the transformation algorithm whilst traversing through a pre-mapped environment.

Table of Contents

WORD COUNT: 8328 WORDS

List of Figures:

List of Tables:

List of Abbreviation:

AT	- Application Transparent
API	- Application Program Interface
UART	- Universal Asynchronous Receiver-Transmitter
I/O	- Input/ Output
RF	- Radio Frequency
FFD	- Full Function Device
RFD	- Reduced Function Device
AL	- Application Layer
ROS	- Robot Operating System
MAC	- Media Access Control Layer
NWK	- Network Layer
AL	- Application Layer
I2C	- Inter-Integrated Circuit
IDE	- Integrated Development Environment
PAN	- Personal Area Network
ID	- Identification / Identity

1. Introduction

1.1 Introduction

Technology in our world is growing faster than ever. Robots are becoming more and more capable of completely complex and time-consuming jobs. Through research it has been found that instead of compressing all the computing power in one unit, distributing the computational power across a system of multiple small robots is more reliable and better way to approach the problem. This concept is known as swarm intelligence (7). Swarm Intelligence can make simple systems much smarter by simply acting as a unit. This idea was originally inspired by nature. Swarm behavior can be observed in most of the animal kingdom. Ants are the best example, a single can only perform limited range of functions, but an ant colony can build bridges and superhighways of food and information, wage war and enslave other ant species (30). Taking advantage of this idea, the project considers autonomous deployment of the pre-existing Robocone project to exhibit swarm behavior such that the system is aware of each other and the dynamic environment.

This report documents the undertaken project in which the autonomous deployment of robotic traffic cone is investigated, which is also referred to as the 'Robocone'. The most efficient and practical ways to carry out the deployment will be researched, exploring the challenges and limitations. Transformation into different formation will be looked into and algorithms for doing so and dealing with the obstacles and path detection will be decided. Simulations will be conducted in order to recognize limitations. Following this, the sensing solution will be investigated with the necessary research and tests conducted. The results will then be used for implementation in robotic-nodes coded to demonstrate navigation through a pre-mapped environment.

1.2 Motivation

Over the years application of multi-agent robots in a co-ordinated and synchronised system has become more popular and demanding. This trend is especially noticed in control, networked operations, sensing and instrumentation to make the take more efficient and effective. (31) Being inspired by complex behaviours in natural swarm system, involvement of multiple robots in activities such as space-exploration, co-operative transportation, environmental monitoring has been increasing steadily [32].

For road maintenance work, a large number of traffic cones is used to redirect traffic to ensure the safety of drivers, so that the maintenance work can be done efficiently and safely. However, the deployment of these cones is currently being done manually; this not only takes up lots of time and manpower but also puts the worker's life at risk. In the UK 26% of total deaths of workers was by a moving vehicle in 2018 (1) with majority of them being road workers.

If this could be done autonomously, it would save time and labour, also ensuring that the job is done with great precision every time by decreasing the chances of any man-made errors that lead to fatal injuries.

Summarizing , it would be very useful to implement an autonomous deployment system for the Robocone system. Which will be remotely controlled through a wireless controller/Server allowing it to communicate within a network, whilst navigating through a pre-mapped environment avoiding the obstacles in doing so.

1.3 Aim

The scope of this project is to develop a semi-autonomous deployment system for the Robocone project enabling it to exhibit swarming and self-organizational behaviour. Allowing multiple robot units to set up in a given formation in the allocated location that is set by a controller and the robots would also be expected to manoeuvre around tricky environment, avoiding obstacles and changing formation to get through small areas.

A viable robot will be selected and then additional sensors along with computations boards will be chosen based on research and simulations. The robot-nodes will be coded in its respective software, alongside the sensor to demonstrate the movement of Robocone system in a pre-mapped environment.

1.4 Objectives

Whilst trying to achieve the aims that has been discussed above, the main would be objectives would be:

- To be able to code in MATLAB to conduct simulation of transformation models.
- To be able to program in Arduino to code the MONA robot-nodes.
- To learn the algorithm of robots to perform swarm interactions.

- Code software for the robot-nodes to show movement and perform transformation when required.
- Establish which sensors are to be used.
- Create an interface between the chosen sensor and the robot hardware.
- To implement a wireless communication system between a server and multiple nodes.
- Optimise software so that the robot-nodes can avoid obstacles by being environment-aware with the help of the sensors and also being able to traverse through tricky paths.

2. Background research and literature review

In this chapter, all the basic concepts and ideas which will be later used in this project will be reviewed. This includes the ideology behind swarm robotics, some analysis and following through all the hardware and software that are used in this project.

2.1 Swarm Robotics

Swarm robotics is defined as a new approach to the co-ordination of robotic systems consisting of large number of relatively simple, small and low-cost robots in terms of their physical body and their controlling behaviour which take advantage of a large population (5). A key component of the system is the local communication between the group enabling it to interact with other robots of the group and also the immediate environment.

Swarm robotics takes advantage of being a decentralized and distributed system.

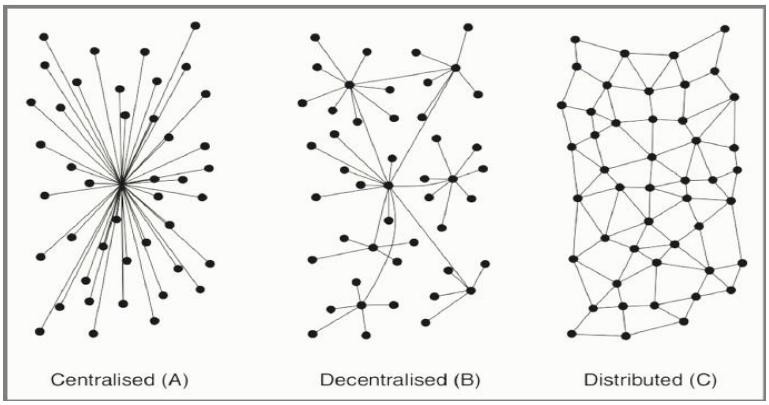


Fig1. Types of Systems (4)

2.1.1 Major difference between Centralized and Decentralized Systems

Centralized systems mean one controller commands everything although it is simple, easy to make and easy to control, if the core controller is damaged or isn't available all the subsystem fails. Also, the central controller is expected to be fast in making decisions from the information

it receives from the slave sensors and provide commands to the actuators which means it is easy to code but quickly gets complicated and expensive as the scaling increases (3).

Decentralized system solves both the problems by having multiple controllers and if failure is encountered in one of the controllers the other parts of the robot are still working. This facilitates easy scalability and addition of new features. It takes longer to design and code this type of system to allow room for abstraction and scalability (3).

2.1.2 Characteristics of swarm robotics (2)

- Compared to a single robot, to complete a sophisticated task, the robot must be designed with complicated structure and control modules resulting in high cost of design, construction and maintenance. Also, a single robot might be vulnerable especially when a broken part may affect the whole robot and it is difficult to debug in such situations. A swarm robotic group will achieve the same through inter group cooperation.
- Parallelism: The size of a swarm robotic group is usually large and it deals with multiple targets at once enabling it to perform tasks that are distributed in a vast environment, saving time.
- Scalable: Since a swarm group is based on a non-centralized system, it allows the individuals to join or quit the task at any time without interrupting the swarm as a whole. This also tells us that the system is adaptable for different sizes of population with minor modifications to the software and hardware.
- Economical: As discussed before the cost of swarm robotics is significantly low and as a whole cheaper that a complex single robot because the individual robotic nodes are mass produced and a single complex system requires precision machining.
- Energy efficient: The individuals in a swarm are small robots which have much lower power consumption when being compared to a single robot. This also means that in most cases they have more battery life meaning they are better in wireless situations.

For the purpose of simplicity and attainability, in this project the network will be setup as a mixture of centralized and decentralized system, where even though all the nodes can communicate with each other, the server is the only one that makes the major decisions based on input it receives from the multiple robotic traffic cones.

2.2 Mathematical Models of Swarm robots

Most swarming systems work on the principle that a plain set of rules at individual level can produce a large set of complex behaviours at the swarm level. Swarm intelligence is the collective behaviour of decentralized, self-organized systems, natural or artificial (7).

One of the simplest models to demonstrate swarm behaviour and swarm intelligence is Boids computer program (6) which works on three principles based on dimensional computational geometry.

These were:

- Separation: steer to avoid crowding local group
- Alignment: steer towards average heading of local group
- Cohesion: steer to move toward the position of local group

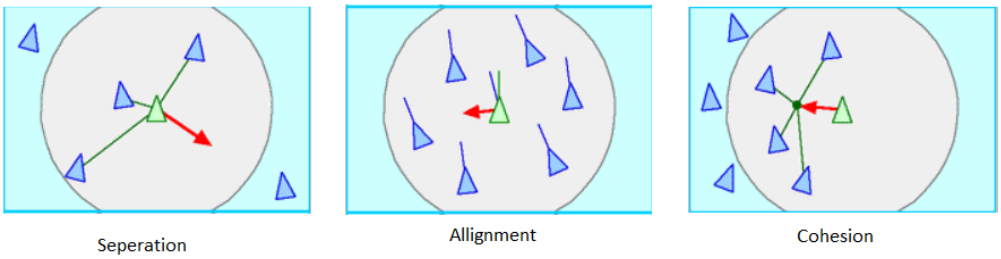


Fig2. Boids swarming model (6)

Inspired from birds, these three rules made sure the group always maintained a minimum and maximum distance between each other while being centred and group whilst moving towards a target.

A slightly more detailed model is discussed by Fredslund, it solves localisation of robotic nodes by simple referencing methods (9). In this model each robotic node has sensing capabilities, a unique identifier and a radio communication model. The ID is broadcasted regularly so each robot knows how many robots are participating in the formation and each robot has its reference partner. By Referencing and positioning itself according to its reference partner each robot can be localized and transformation can occur with simple commands.

In this project sensors will be selected to perform obstacle avoidance which will also be used for localisation which will be discussed later in the System Design Chapter.

2.3 Environment Analysis

The Robocone Project is set to run on the roads and help blocking the area that require to be blocked for construction/repair work. This environment is very accident prone (8), rough and dynamic. The slightest of errors could easily result in huge accidents because of fast moving vehicles. This is very challenging to deal with, later in this report it is discussed how this problem is dealt with.

This project assumes that the roads are pre-mapped with respect to hardware and software, which also means that the mission is predetermined. However, roads are not always obstacle free and the paths are tricky. The Robocone system needs to deal with these kinds of situations and avoid stray obstacles, some examples are shown below. Robocone should be able to make smart transformations according to the path which will be detected by one of the robotic node's sensors and communicated to the rest of the swarm to carry out the transformation to the respective formation.

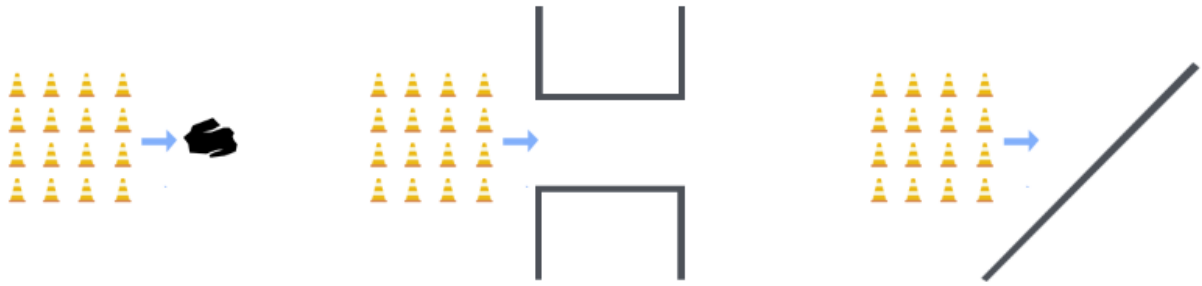


Fig3. Examples of obstacles for the proposed system

2.4 Wireless Communication Model

To support the system developed in this project, the wireless communication system will require one server unit communicating with multiple Robocone nodes wirelessly. The server receives sensor status information from different nodes, computes it to decide which working 'Mode' it should operate in and informs all the nodes in the network to update its working 'Mode' if necessary. This can be done using Bluetooth or Zigbee architecture. In the next section, which one is the most suitable for this project shall be discussed.

2.4.1 Comparison between Zigbee and Bluetooth:

The two most common wireless personal area network (WPAN) standards established by the institute of Electrical and Electronic Engineers (IEEE) under IEEE 802.15 are Bluetooth (IEEE 802.15.1) and ZigBee (IEEE 802.15.4).

Shown below is a spec comparison between Bluetooth and Zigbee. Zigbee architecture protocol is selected as it is a more suitable option to generate a good communication network after comparing factors such as range, length of messages, propagation time, area and interference.

Table 1.Comparison of Bluetooth and Zigbee (10)

	Bluetooth (IEEE 802.15.1)	ZigBee (IEEE 802.15.4)
Representative	HC-05 (Bluetooth 2.0)	XB24-AWI-001 (XBee Series 1)
Range	10m	30m
Networking Topology	Ad-hoc, Piconet, Scatter net	Ad-hoc, P2P, Star, Mesh
Operating Frequency	2.4 GHz	868 MHz (EU), 915 MHz (US & Aus.), 2.4 GHz (worldwide)
Power Consumption	Supply Voltage: 3.3V-3.6VDC Working Current: <50mA Transmit Power: +4dBm	Supply Voltage: 2.8-3.4VDC Transmit Current: 45mA@3.3VDC Receive Current: 50mA@3.3VDC Transmit Power: 1mW(+0dBm)
Modulation Technique Spreading	Frequency Hopping Spread Spectrum (FHSS)	Direct Sequence Spread Spectrum (DSSS)
MAC Scheme	FH-CDMA	CSMA-CA
Protocol Stack Size	250 Kbyte	28 Kbyte
Data Rate	1 Mbit/s	250 Kbit/s (2.4GHz band)
Latency	3-10s	30 ms
Number of nodes per master	7	64K
Cost	Low (Up to £7)	High (Up to £20)

The Zigbee architecture can perform broadcasting and allow scalability to perform star network topology that is required in this project. Since Bluetooth takes longer to perform point-to-multipoint communication through piconet as master only connects to one slave device at a time (10). Zigbee also works with low data rate, has low power consumption and has intermittent data transmissions for control purposes. Zigbee also supports immediate join time and does not require unobstructed line-of-sight and less complexity as compared to Bluetooth. After examining the findings, ZigBee is considered a better option for this application.

2.4.2 Application Layer Zigbee

This project will only involve making modifications to the Application Layer (AL) of the Zigbee. Application Layer (AL) contains the application that run on the network that is used by the user (29). The Physical Layer (PHY), Media Access Control Layer (MAC) and Network Layer (NWK) will be treated as a black box in this project.

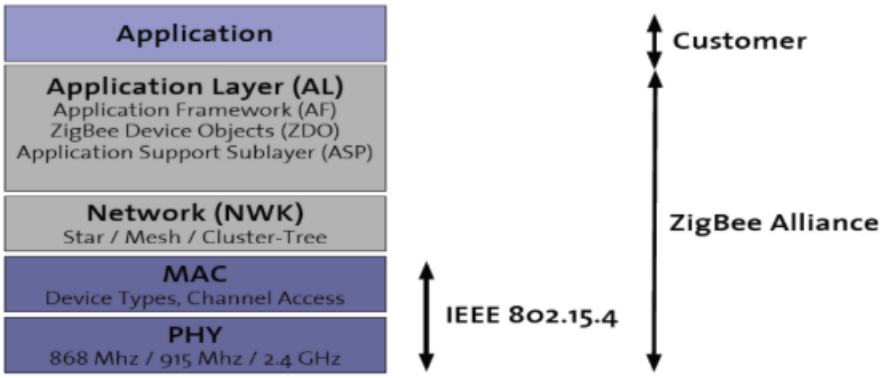


Fig4. Zigbee protocol stack (9)

A node can set to be able to perform all tasks and operates in the full set of IEEE 802.15.4 MAC layer RFD with limited tasks only. The three Zigbee device types (28):

- End device: (RFD) – It can perform data transmission by sending and receiving messages. However, it needs co-ordinator or a router as a parent device to be able to join a network and store messages when in sleep mode. It sleeps most of the time and wakes up briefly to stay in the network.
- Co-ordinator: (FFD) – Only one co-ordinator exists in the network and is responsible for overall network management. Forming the network, assigning how addresses are allocated to nodes and allowing routers or end devices to join the network. It does not sleep.
- Router: (FFD) – It can receive or transmit data and find route to the destination while performing similar functions to the co-ordinator. It also does not sleep. There can be multiple Routers in the same network.

The system in this project will be setup with one server which will be setup as a coordinator and multiple robotic nodes that are setup as routers/end-points.

2.5 Background of MONA educational robot

The MONA robot is an autonomous mobile robot developed at the University of Manchester Robotic Lab (14) in School of Electrical and Electronic Engineering used for demonstrating swarm behaviour for educational purposes.

MONA is used in this project rather than other mobile robot platforms such as Jasmine, E-puck and Kilobot mainly because of availability reasons and also because it is a modular robot that allows easy deployment of RF module for communication with the main Arduino processor via I2C, SPIC and RS232 (14).

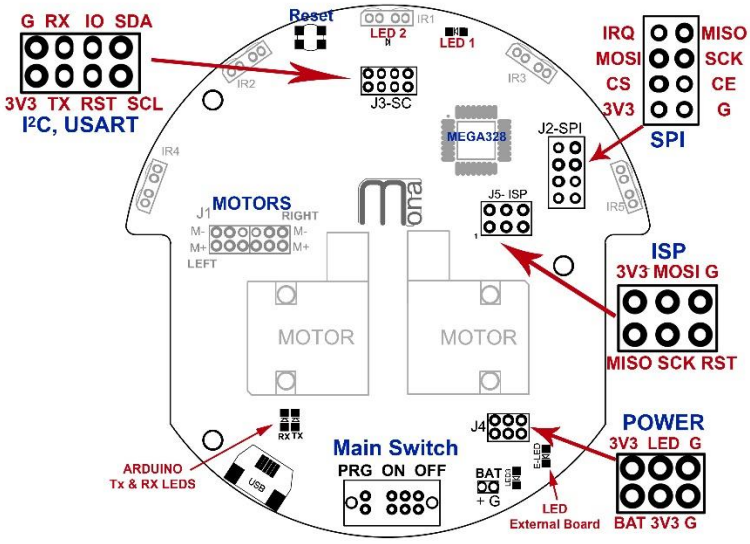


Fig5. MONA-Pin-Config (14)

The AVR 8-bit microcontroller (ATmega 328) with Arduino Pro or Pro Mini architecture is the main processor on the MONA robots. It also uses pulse width modulation (PWM) approach to control the rotational speed of each motor individually (17).

After conducting tests for the five proximity sensors already present on the MONA it was found that the range of them is <2mm and is not sufficient for the purpose of this project. Alternative solutions had to be considered. After exploring other sensing options, it was decided that the HC-SR04 ultrasonic sensor is the best solution because it is cheap and has a 2cm – 500cm sensing range with a effectual angle <15deg (19).

To implement the new sensor-array, an additional Arduino UNO board had to be used due to insufficient connection ports in MONA robots to interface the HC-SR04 sensors. This board connected to the MONA robot using the I2C interface via UART.

3. MATLAB Simulations

The robotic system will have to traverse through the mapped environment using different formations. Many different formations were compared and it was found that a N x N (square) formation is the most compact and efficient formation to have as a base or starting formation. To travel across the planned path, the formations will have to constantly change. Using MATLAB different combinations of steps are compared to perform a transformation 4x4 formation to a 2x8 formation to a 1x16 formation. This is discussed in detail in Transformation model (4.2.4) section of this report.

3.1 MATLAB Program

Based on the Transformation Model in 4.2.4 a program in MATLAB was created to demonstrate the transformation models to perform obstacle avoidance and final mission with the help of MATLAB's plotting tools. The screenshots of the below illustrate few stages of the MATLAB simulation.

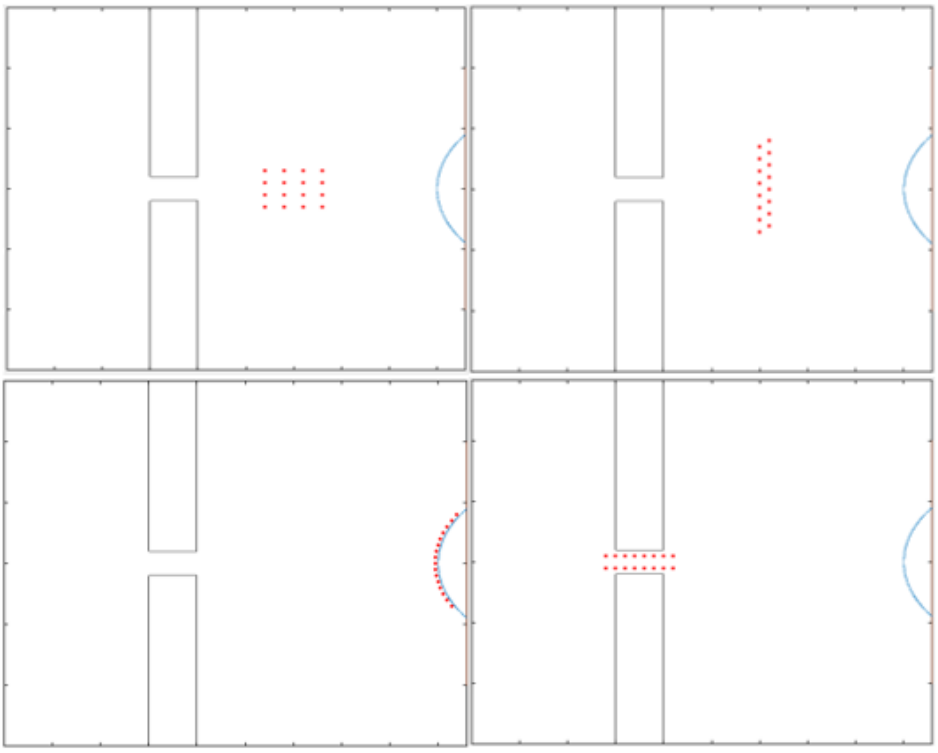


Fig6. MATLAB Simulation Screenshots

4. System Design

The main components used in this project are the MONA educational robots, XBee series 2 modules, XBee development boards, HC-SR04 proximity sensors and Arduino UNO boards. Software tools that will be used are Arduino IDE for the MONA robot's and the Arduino UNO

boards and XCTU for XBee configurations. This chapter covers the hardware and the software design of the proposed system. Also going through the implementation of Transformation models and Wireless Communication System.

The Arduino UNO board is chosen due to its small size, low power consumption, wide pin selection used to interface the proximity and e-compass sensors to the MONA acting as a computation board.

4.1 Hardware Design and Implementation

The hardware design is not too complicated and easy to implement. By using lots of jumper cables, the connections between the different components were made for testing purposes. Later, a PCB was implemented for better and secure connections. Only 4 pins were used to connect the XBee modules, as digital line passing is not used in this project.

4.1.1 XBee, XBee Development adapter and X-CTU

Zigbee that is based on IEEE802.15.4 with low data rate and intermittent data transmissions is selected for monitoring and control purpose as the most suitable option as the project involves short range communication network.

According to Digi XBee 802.15.4 official website the low cost, easy to deploy RF modules can be used as a serial cable replacement for serial communication or as a part of a complex system. The XBee modules which support ZigBee protocol make point to multipoint communication easy and possible with an unobstructed line of sight, less complexity and flexible radio frequency (RF) communication.

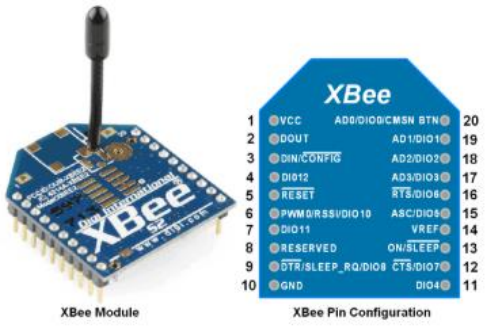


Fig7. Pin configuration of an XBee module (10)

As seen in the figure 7 each XBee module has 20 pins, however only four main pins are used in this project: VCC (Pin:1), DOUT (Pin:2), DIN (Pin:3) and GND (Pin:10). Data will enter from the Arduino using the DIN pin and exit the module via the DOUT pin. Devices with UART interface

(MONA and Arduino UNO) can directly connect to the transmitter and receiver pins of the radio module for serial communication using the RX and TX pins, provided that both the connected XBee RF modules have been configured with compatible settings using the XCTU software.

The X-CTU software is an easy way to configure these RF modules and uses the same settings to communicate with an XBee modules. The settings used are:

- Baud = 9600 bits/second
- Flow control =None
- Data bits = 8
- Parity = None
- Stop bits = 1

Fig8. XBee THT Groove Development Board (10) ->

PC's do not have ZigBee capability. Also, XBee modules require 3.3V power supply and have 2mm pitch for the pins. Most microcontrollers usually have 5V logic signal. The XBee development board solves both issues by regulating to 3.3V providing an easy way to connect to the PC for configuration purposes (11).

X-CTU is a software tool provided by Digi International to configure the operation mode or device function type. It also provides a way to test device performance and upgrade the firmware of the XBee modules. It is very important to make sure that the XBee modules are correctly configured so that wireless communication can be carried out successfully within the same network. The important configuration settings for the XBee series 2 for network communication being ID (PAN ID), DH (Destination Address High), DL (Destination Address Low), MY (16 Bit Source Address), CE (Coordinator Enable), AP (API Enable).

PAN (Personal Area Network) ID: This is unique for each ZigBee network. All ZigBee devices configured with the same PAN ID will join the same network, set to 3332 by default (12). There are two addressing method which are short (16 bit addressing mode) and long (64 bit addressing mode). In this project for simplicity short 16 bit addressing mode is used by all the XBee modules.

The NI parameter represents the node identifier is used to differentiate multiple RF modules by naming them. In this Project, XBee which acts like the coordinator is named SERVER and the

rest are named NODE. The CE parameter is used to set if the RF module will act as a coordinator or end-device by setting to 1 or 0.

Source address and destination address that are denoted by MY and DH or DL respectively have been set for every XBee module as per requirement.

Table 2: Configuration Settings for XBee Modules in this project

	NI	DH	DL	CE
XBEE 1 (SERVER)	SERVER	0	0xFFFF	1
XBEE 2 (NODE)	NODE	0	0	0

After configuration of the radio modules respectfully the radio modules will be in the same network and can be shown as illustrated as shown in figure below. It is a bidirectional network and the end-devices send data only the co-ordinator and the co-ordinator broadcasts the data. This is done by setting the 16-bit destination address (DL) to 0xFFFF for broadcast and 0 for send to co-ordinator.

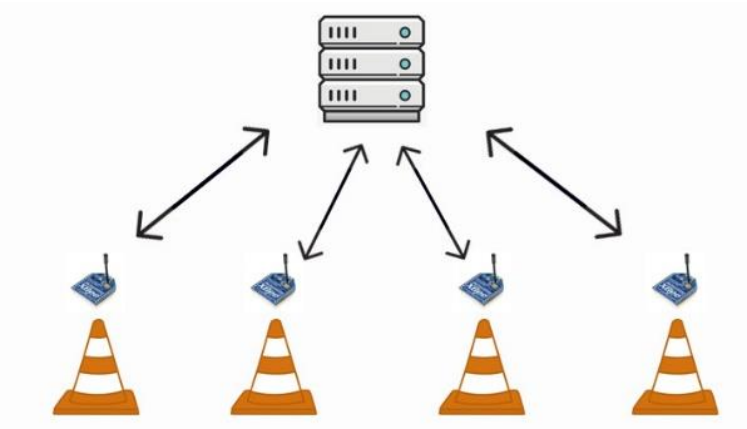
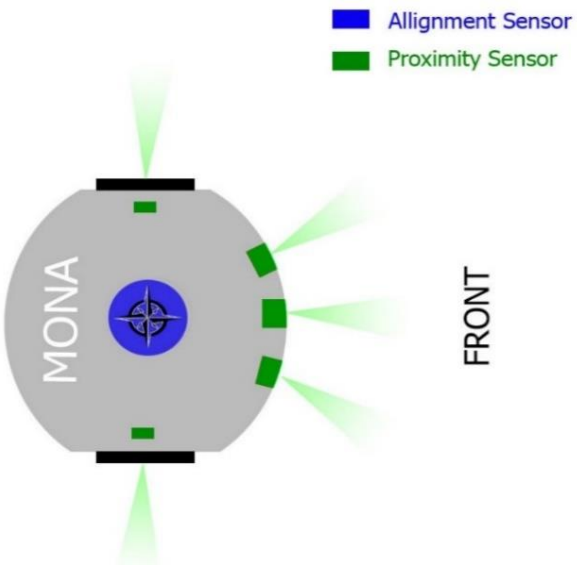


Fig9. Star topology network

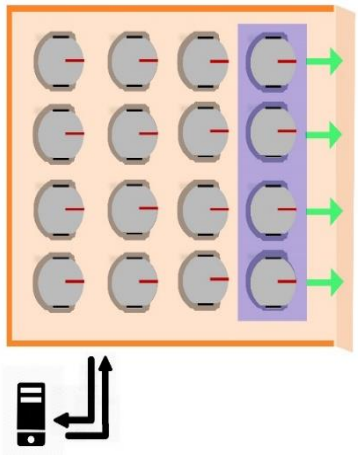
4.1.2 Sensor Layout and Working Ideology

For Sensing requirements of this project, this plan was first proposed. In figure 10, It can be seen that it will require 3 proximity sensor in the front of the robot and two at the sides to check for clearance in each direction. It will also require an e-compass sensor to check the



alignment of each robot with its neighbour at each side and at the rear. Fig10. Sensor layout →

Working Ideology:



Below, the proposed idea for the working of the system and how the data from the sensors will be used to complete the desired mission is analysed.

This figure shows an example of the base station idea. The robots will start from a fixed position with the help of a base station. So, the starting position is known and the displacement of each robot from every other is also known. ← Fig11. Base station representation

All the nodes are connected to a server (laptop/PC) which they communicate with, stablishing a star topology network for the system.

The four robots highlighted in blue are considered as the leaders as they have extra functions to do. However, the leaders are just a concept and is only fixed for a given formation. This means leaders change when the formation changes.

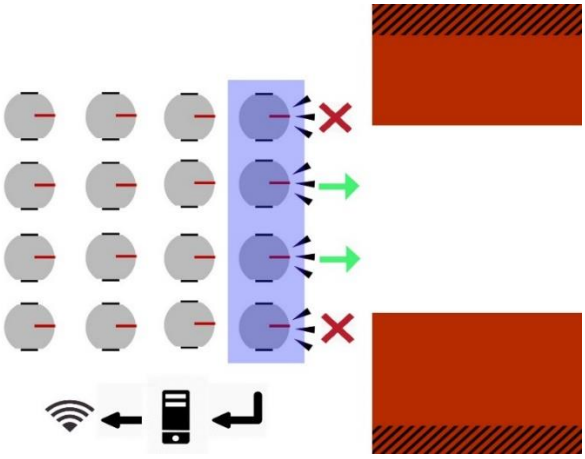


Fig12. Situational Example 1

When encountering a narrow path, similar to the one shown above. The proximity sensors of leader 1 and leader 4 (leader 1-4 numbered top to bottom in column 4) say that the path is blocked, however the proximity sensors of leader 2 and leader 3 say that the path is clear. This will be informed to the server and the server will command the rest of the system to perform a transformation to a 2x8 so it can traverse through the narrow path.

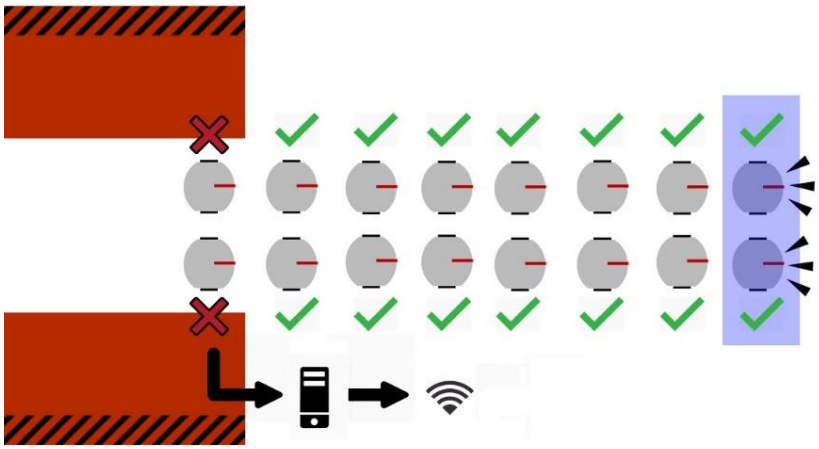


Fig13. Situational Example 2

The proximity sensor located on the left and right of each Mona robots assist the system in checking for the end of the narrow path. In the illustration above, the last pair of Mona robots are telling the server they still are in the narrow pathway. When it surpasses it, the server is notified, and it is free to perform transformations if it is necessary.

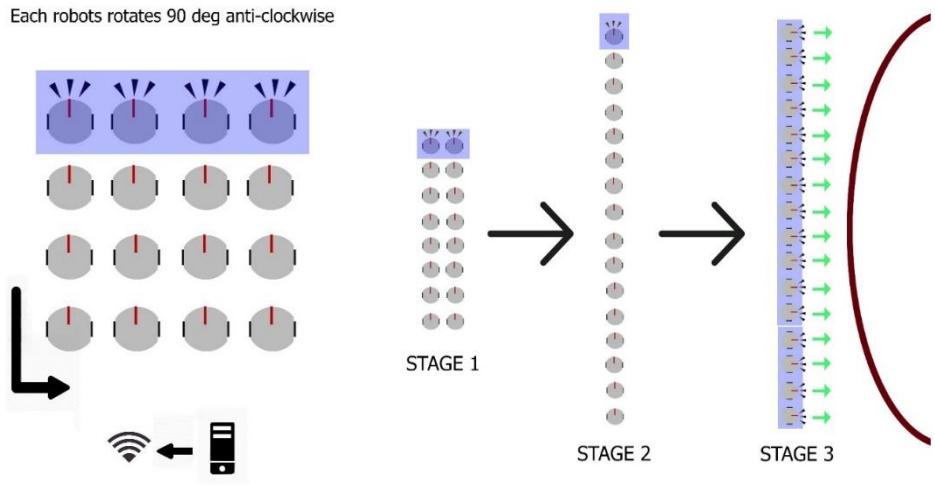


Fig14. Situational Example 3 & 4 (a & b)

Figure 14(a) demonstrates, how rotation commences in the group leading to a change in the leaders accordingly. After the rotation the top four are the leaders and actions will take place with respect to those four robots. *(the leaders are highlighted in blue).*

The final stage, Figure 14(b) of every mission is shown above where the following transformation (4x4 -> 2x8 -> 1x16 -> rotation by 90 deg) takes place and with the help of the proximity sensors the robots approach and cone off the assigned area.

However, in the practical implementation of this project only three proximity sensors (front, left and right) were used and the rest (e-compass and additional proximity sensors) were omitted due to cost and time restrictions.

4.1.3 Wire connections of Arduino UNO with MONA, XBee and Proximity Sensors:

The Arduino UNO links the three different parts of the Node together the 5V pin on the Arduino UNO board is used to power the proximity sensors and 3.3V pin is used to power the XBee RF module. The Echo and Trig pins of the three proximity sensors are connected to six digital IO (13-8) for reading data from the sensors. The Arduino architecture performs serial communication with the RF module through the UART pins. As shown in the figure below the pins required to send and receive data are RX, TX, 3.3 and GND. The MONA Robots have built in battery acts as the source of power to the MONA robot and the Arduino UNO board is power with the help of a rechargeable 9V, 200mAh NiMH battery. All of them are connected to a common ground connection for correct operation.

The SCL of the MONA is connected to the SCL of Arduino UNO board and similarly the SDA of the MONA is connected to the SDA of Arduino UNO board to perform serial communication between the two boards using I2C interface.-

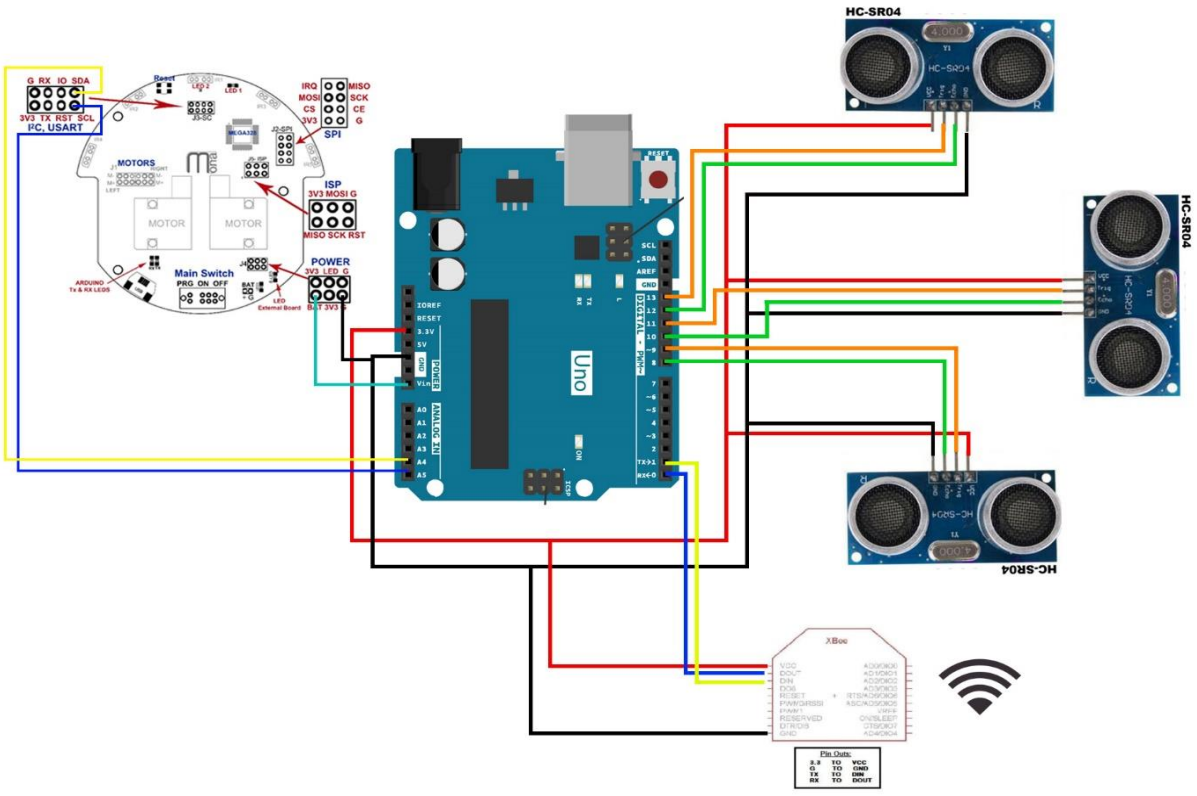


Fig15. Connection Layout

The right side of the schematic above is made into a PCB to make the connections clear and secure to minimize the probability of a loose connection or a short circuit. The PCB is design in such way that it plugs into the Arduino UNO board and the Arduino UNO board is taped onto the MONA robot.

4.1.4 PCB Design

A PCB was necessary for this design as the XBee and the sensor had to removed and connected often as uploading code onto the Arduino Boards is not possible when the RX and TX port of the Arduino are connected to the XBee RF module.

Using headers, PCB connectors and soldered wires the connections are made between different parts as shown below.

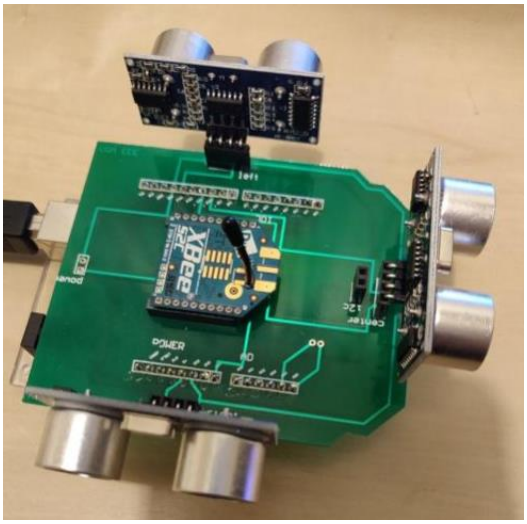


Fig16. PCB Connections →

4.2 Software and Theoretical Development

4.2.1 MONA educational robot and Arduino UNO

The MONA educational robot and the Arduino UNO board with the Arduino architecture uses Arduino's C/C++ derived programming language to implement the software for motion control / transformations / obstacle avoidance. The library that must be included and installed before the compilation of the code are MONA.h, Wire.h and TIMER ONE library.

Through the Arduino IDE, some configurations must be done before connecting and programming the MONA robot. The settings required are (16):

- Boards: "Arduino Pro or Pro Mini"
- Processor: "ATmega328 (3.3V, 8MHz)
- Programmer: ArduinoISP.org
- Bootloader: ATmegaBOOT_168_atmega328_pro_8MHz.hex

Similarly, to configuration settings for the Arduino UNO board are:

- Boards: "Arduino Genuino/UNO"
- Programmer: ArduinoISP.org

- Bootloader: ATmegaBOOT_168_atmega328_pro_8MHz.hex

4.2.2 Code and role of each component

Figure 17 is the representation of a single node with the XBee, MONA and UNO board labelled respectfully. In this section the working of the code in each section is discussed.

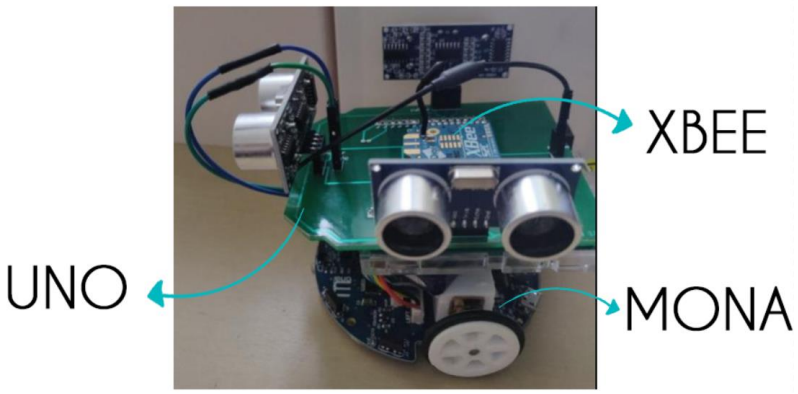


Fig17. Robotic Traffic Cone NODE

Arduino UNO:

The implementation of the Arduino UNO board here is to provide enough connection ports and to simply act as computing unit. It also makes communication between the server and the MONA robot possible. The Main purpose of the Arduino UNO board to get receive the sensor data, compute it to find the sensor status and send it to the server and receive data from the server and updated it across the whole node (UNO + MONA).

Figure 18 (page 19) shows the program flowchart of the Arduino UNO code during execution. In the initialisation stage, all the pins that are required for the sensors and the XBee connections are initialised along with a Software Serial called XBee defined on port (0,1) and a char variable called ‘Mode’ initialised with a value of ‘0’. ‘Mode’ acts as a select condition in the code.

Outside the structure of main loop, functions check_forward(), check_left() and check_right() are defined, which each return the distance from the closest object when called. Additionally, a function named check() is defined which computes sensor status from sensor values by taking in 3 inputs (closest distance from the obstacle in left, right and centre direction) and returning a Boolean value which indicates the conditions for the obstacles being in range.

In the main loop with the help of a switch select code using conditional (if, else) statements, one of the 5 loops with the value of ‘Mode’ as the select condition is selected. In each loop, the sensor values are obtained if required and passed through to the check function, the return

value of the check function is sent to the Server using function Serial.print(). If the Arduino board receives input from the XBee (checked using XBee.available()) device then 'Mode' is updated using function XBee.read() to the value that is received, if nothing is received the loop continues with the unchanged 'Mode' value. At the end of each loop using wire.write() it updates the value of 'Mode' on the MONA which is connected using UART.

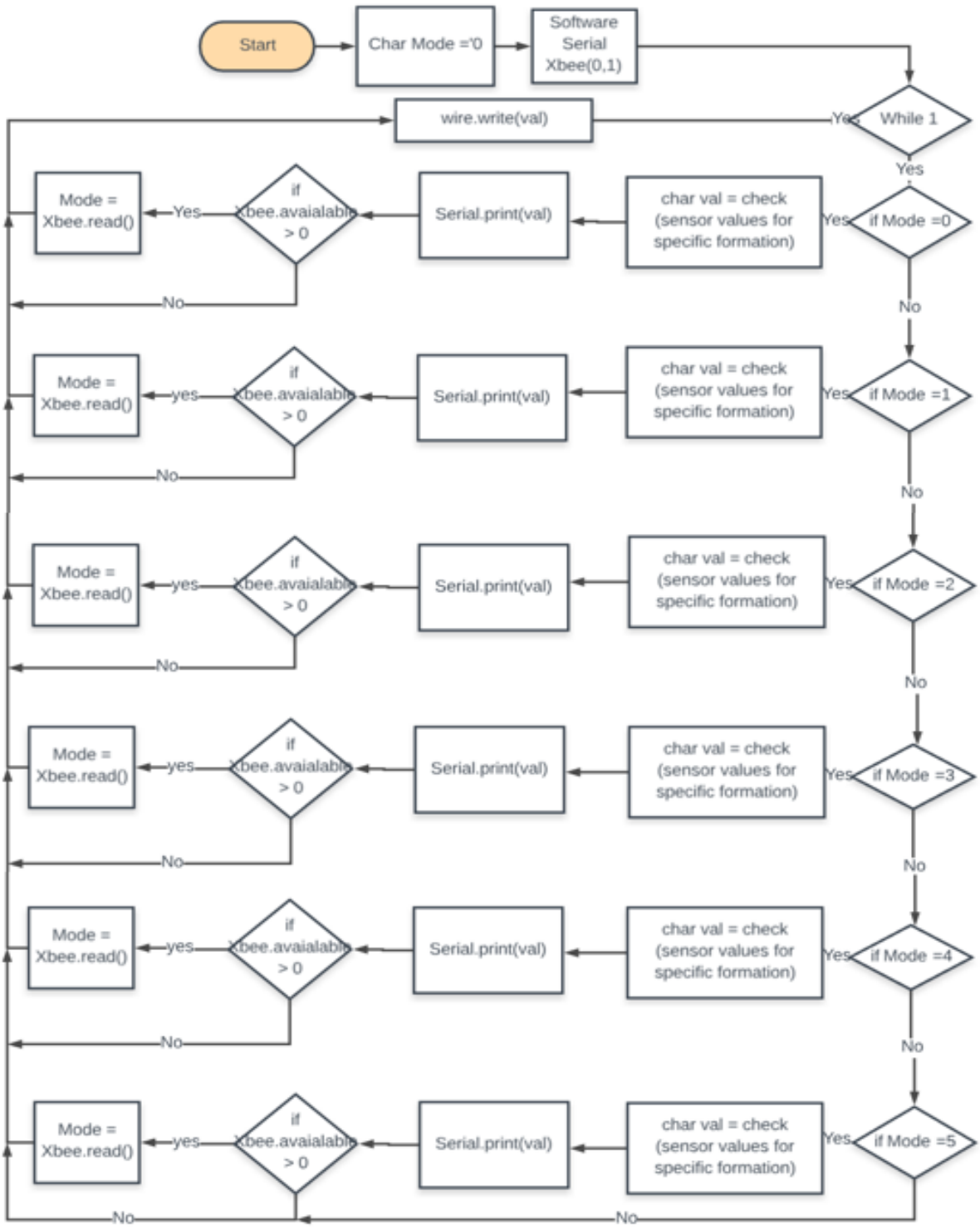


Fig18. Flowchart of Arduino UNO

Mona Educational Robot:

The sole purpose of the MONA Educational robot in this project is to perform motor control. This code also works on a 'Mode' select basis as well. It receives which mode to run on using the I2C interface via UART from the UNO board.

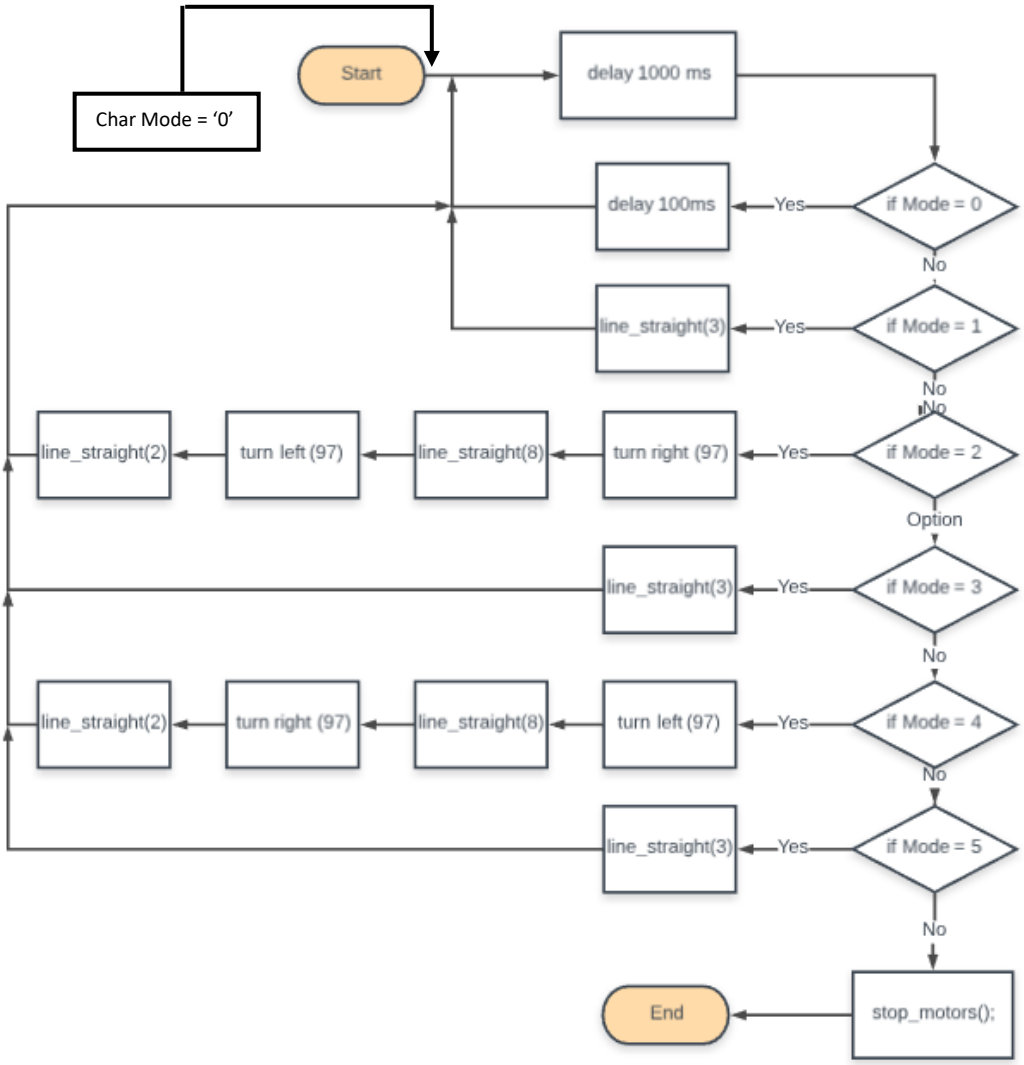


Fig19. Flowchart of MONA Code

Using the encoder on the MONA and the implemented PID control, functions for moving the robot 'x' distance straight, turning 'x' degrees left and right is defined outside the main body of the code. The receive_event() function which is called everytime it receives a value via the UART and updates the value of 'Mode' with the received value every time it is called. The main loop selects which sequence to run based on 'Mode' and terminates when it reaches 'Mode' 6.

Note: Figure 18 and 19 are unique to one of the nodes (left front in this case) because the combination of sensors and motors movements are different for each node.

4.2.3 Wireless Control System (Server)

The role of the Server is retrieving the sensor status from all the different nodes and if a change in formation is necessary then it updates to the next operating mode and broadcasts this across the whole system. This is done with the help of the code shown in the form of a flow chart below.

In this project one of the older version MONA is used as the Server in which a single XBee configured as Coordinator is connected to the Arduino using the RX, TX, 3.3 and GND Pins.

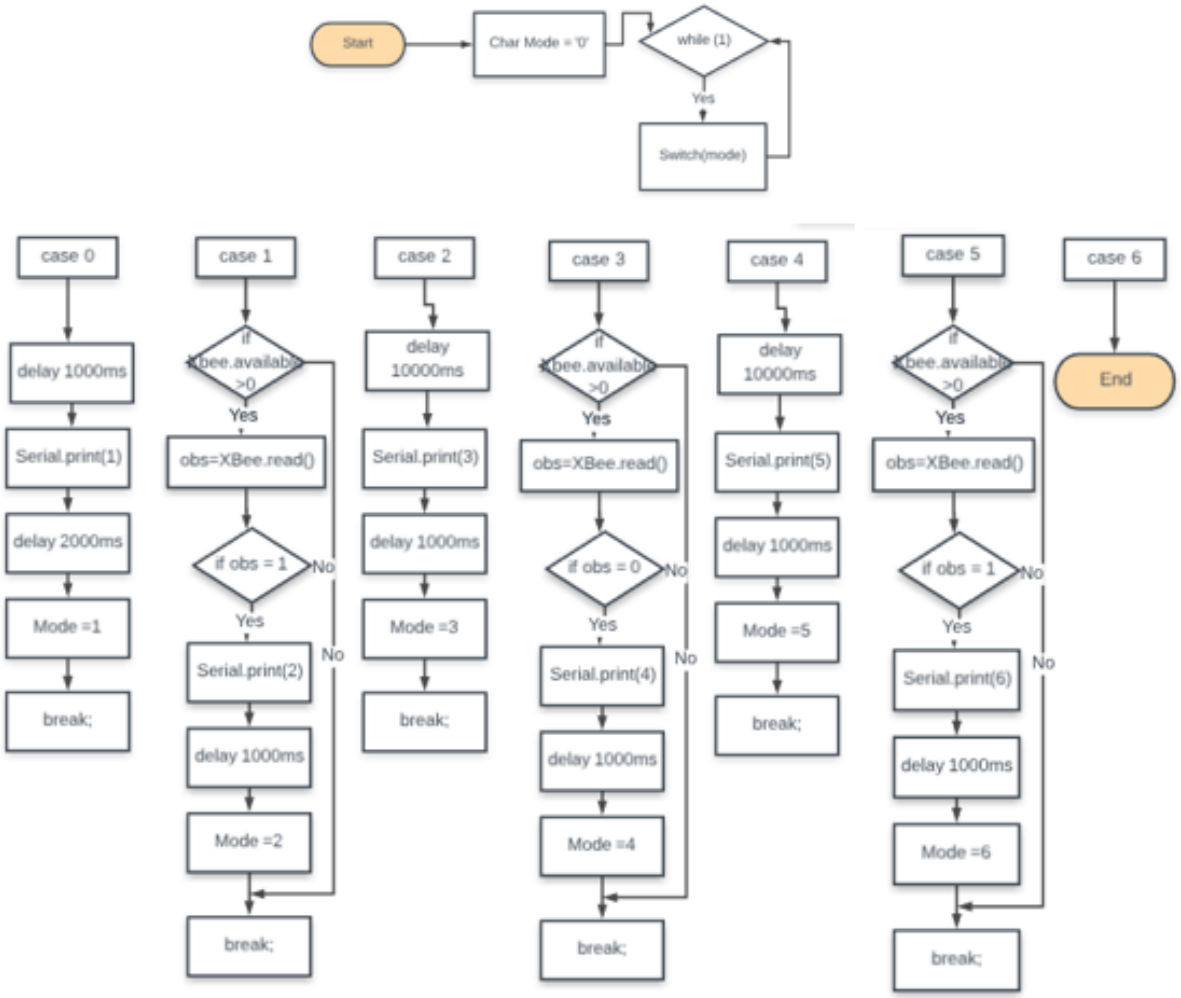


Fig20. Flowchart of SERVER

4.2.4 Transformation model

For the proposed system to work i.e. the robotic nodes to cone off a certain predetermined are of the road or construction site. The Robotic system will have to follow certain formations.

It is decided that for research and development process sixteen robot-nodes will be used to represent the system however this system will easily be scalable up to whatever number of robotics cones are required for the operation.

These 16 robotic nodes will have to be deployed off a vehicle or a station. A 4x4 square formation is decided as the best option for the initial deployment stage. It is the most compact formation where the nodes maintain a minimum distance from each other, this formation also enables the robotic nodes to be charged wirelessly through inductive charging in a charging station which is represented in the figure below.

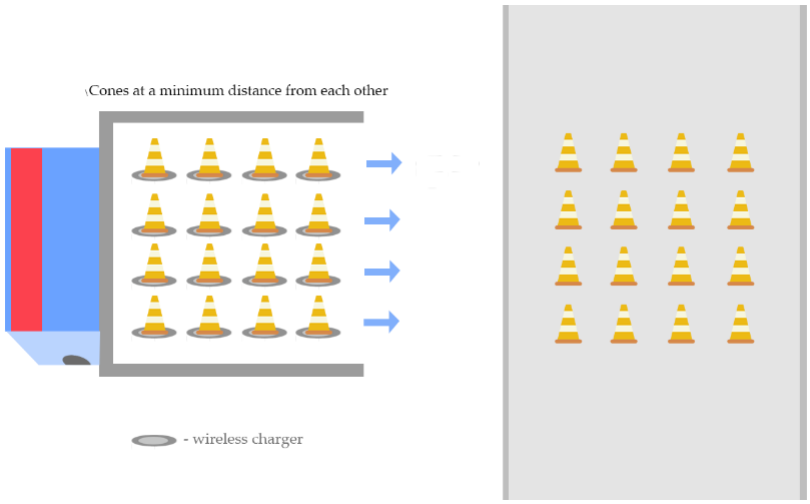


Fig21. Wireless Charging Representation

These 16 robotic nodes will have to undergo various transformations to maneuver challenging pathways with the basic transformation being changing to a double file formation and then two a single file formation. Considering maximum area required and the number of steps required to carry out the transformation it was decided that this will be done by interlocking the inner columns with the outer columns.

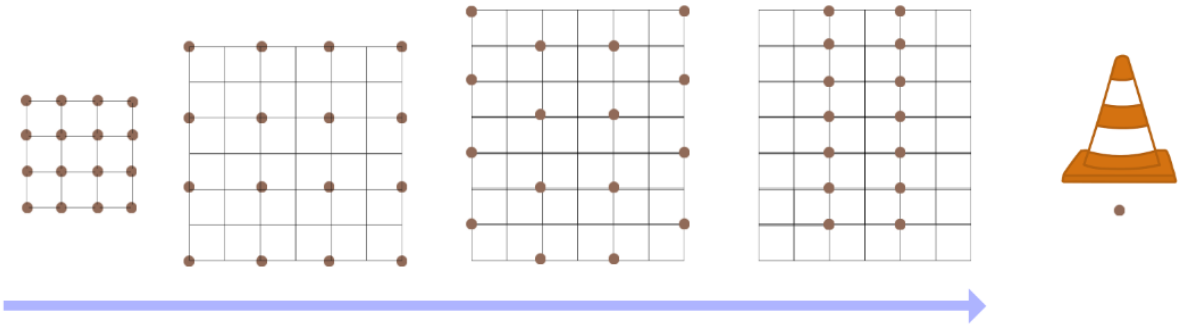


Fig22. Transformation to a file of two

To achieve this, the grid expands and then the first and last columns move upwards and move towards the center merging into a file of two as shown in the figure. Similarly, this file of two will transform to a file of one by interlocking of the two columns. This sequence is easily reversable for retrieval of the system back to base.

The final stage of every mission will be to enclose the designated area in cones. This will be done with the help of the sensors available on the robots. The robots will move towards the target once they are in a single file formation placed perpendicular to the target. With proximity sensors and radio modules to communicate with other members of the group the robots will maintain a fixed distance and a maximum angle with each other while trying to achieve a minimum distance with the target.

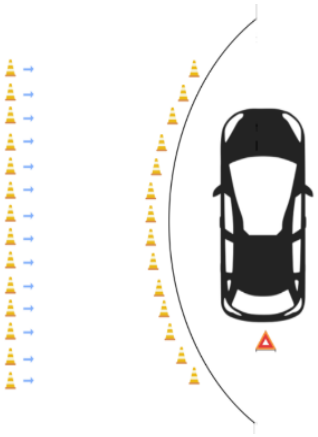


Fig35. Target approaching ideology →

5. System Testing

Due to lack of resources and time, the project experiments were conducted with only four robotic traffic cone nodes and one server, However the whole concept still remains the same.

5.1 HC-SRO4 Proximity sensor testing

The HC-SRO4 ultrasonic sensor uses sonar to calculate the distance to an object similar to what bats do. It offers excellent non-contact range detection with high accurate and stable readings. It is simple to use and easy to implement.

5.1.1 Implementation

To test the sensor, The Arduino is programmed to turn on and off the trig pin of the HC-SR04 really fast (10 microseconds) which makes the transmitter in the sensor send a signal (high frequency sound) and then it waits for the signal from the receiver that picks up the reflected high frequency sound (20). From this the time between transmission and reception of the signal is known and the distance in cm is calculated using a function called microsecondsToCentimeters(long microseconds). This function calculates distance in cm by using the equation: *distance in cm = microseconds / 29 / 2* .

The connections required for the operation of the above-mentioned ultrasonic sensor are VCC -> 5V, GND -> GND, Trig to Pin# and Echo to Pin#. The test is conducted for distances 1-10 cm's and then at 30cm.

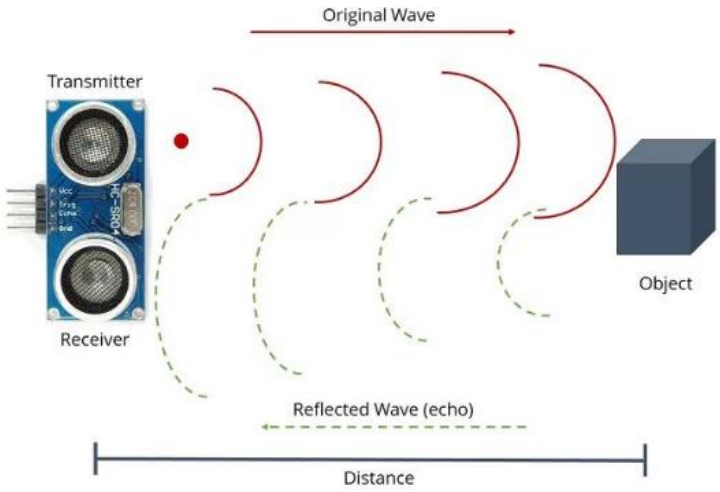


Fig23. Working of HC-SR04 ultrasonic sensor (20)

Y-Axis Test: Another one of the major problems with the application/use of this ultrasonic sensor as a proximity sensor is that the Sensing is done in 2-axis and a constant Y-axis. This means that if the Sensor is located at height 'h' then it can only detect obstacles at height 'h' with some margin up and down say 'x'. This test is to calculate the x value.

The Sensor is wired as mentioned previously and an object is moved up and down to calculate till which height the object is detected by the ultrasonic sensor.

5.1.2 Result Analysis and Discussion

After making necessary connections the accuracy of the HC-SR04 proximity sensors is tested and the results are shown below.

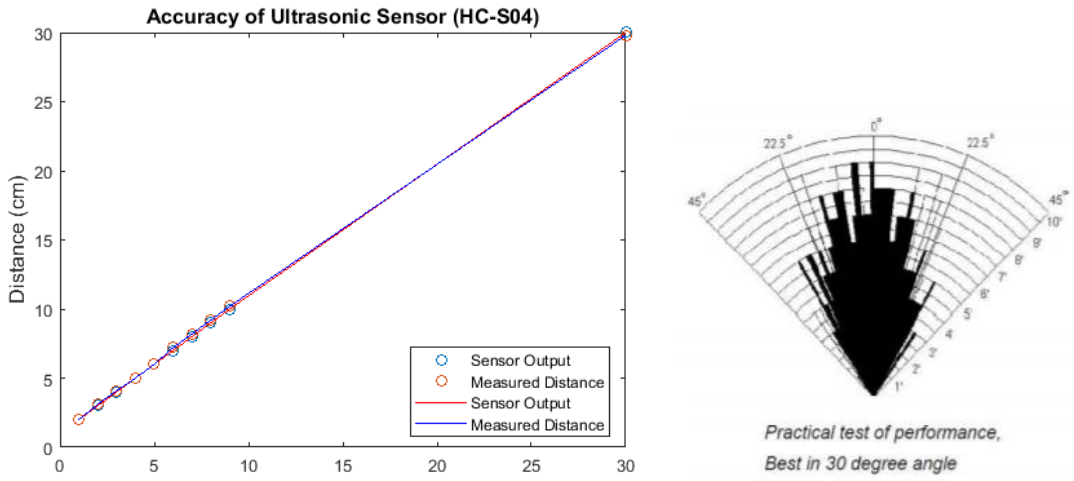


Fig24. Comparison between Actual distance and Sensor Output and Cone characteristic of Sensor

As it can be seen the HC-SR04 proximity sensor is very accurate with minimal error (+- 2mm max) between actual distance and Sensor output especially in the operating range that is required for this project.

As for the cone behavior of the sensor, it is advantageous that in close range it does not pick up objects/obstacles in the horizontal axis however same can be said in the Y-Axis where it doesn't pick up the signal if the object/obstacle is higher or lower than the level that the sensor is setup + a margin 'x' which is found to be ~ 1-2cm when the obstacle is at 5-10 cm distance.

5.2 Communication

There are multiple levels of communication required in this project, all of the them are discussed below.

5.2.1 Arduino Communication

The communication between the MONA robot and the Arduino UNO board is a wired link. The exchange of information is done using the UART ports using I2C protocol.

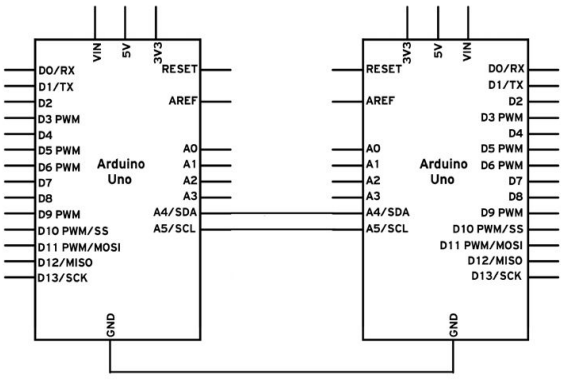


Fig25.Arduino to Arduino connections – I2C (21)

Implementation

To implement the exchange of information between the two boards. Simple tests are conducted between two Arduino Uno boards. The connections shown in figure 25 are made. It is necessary to include the wire.h library in the code. This gives access to functions such as wire.write() and wire.read()

Test: The Arduino on the left (master) is setup on address 9 using wire.begin(9), it then writes the value of 'Mode' onto the address '9' using Wire.beginTransmission(9) followed by Wire.write(x); and Wire.endTransmission(). The Arduino on the right (slave) is setup in a similar way and is programmed so that a function recieveEvent is called every time something is received using void Wire.onReceive(receiveEvent) and this function (receiveEvent) assigns the received value to some local variable, 'Mode' in this case.

Result Analysis and Discussion:

The setup of serial communication between two Arduino chips is straightforward, worked as intended and was easily setup using the I2C interface.

5.1.2 Point-to-Point and Point-to-Multipoint communication – XBee

The communication between the multiple nodes is a wireless link between the XBee devices. XBee devices communicate by exchanging information via modulation of waves in electromagnetic spectrum with the same frequency (29).

XBee devices can run in two modes namely AT mode and API mode, this is selected by setting API to 1 for API mode or to 0 for AT Mode. The main difference between the two is that in API Mode the XBee device receives information in terms of data packets (13). This has multiple advantages. Some of them are that the configuration of the XBee devices can be set and read within the network, data can be transmitted to multiple destinations and the received data includes transmission detail such as reason for success/failure and includes the sender's address. Comparatively AT/Transparent mode is simple, direct and easy to use.

In this project, AT Mode is used for simplicity as only continuous update of which 'Mode' the system should be running is required which should be common throughout the system.

Implementation:

Multiple Tests were conducted to before setting up the wireless system between the different nodes and the Server.

- Test A: XCTU Software (AT – Mode) point-to-point

  The first test conducted will be setting up a wireless chat system between two XBee Devices connected to the PC using the Development boards. In the XCTU Software the configuration is setup as shown in table 3. DL is set to '0' to send data to the coordinator.

Table 3: Configuration Settings for XBee Modules in Test A

	ID	JV	DH	DL	CE	API
COORD (TX)	9999	0	0	0xFFFF	1	0
END(RX)	9999	1	0	0	0	0

- Test B: Arduino and XCTU (AT-Mode)

  One of the XBee devices configured as END or NODE is removed and is connected to Arduino UNO board is similar way as seen previously (TX → RX, RX → TX, 3.3V → 3.3V and GND → GND) The code that outputs the sensor status on the UNO board that is connected to the XBee RF module along with the HC-SR04 ultrasonic sensor is run and the results are observed after opening the port on the Console Window on XCTU. Since the XBee is connected to the Serial port (RX, TX → pins 0,1) of the Arduino UNO board, functions like Serial.write(), Serial.read(), Serial.available() are used to transmit and receive data on the Arduino side.

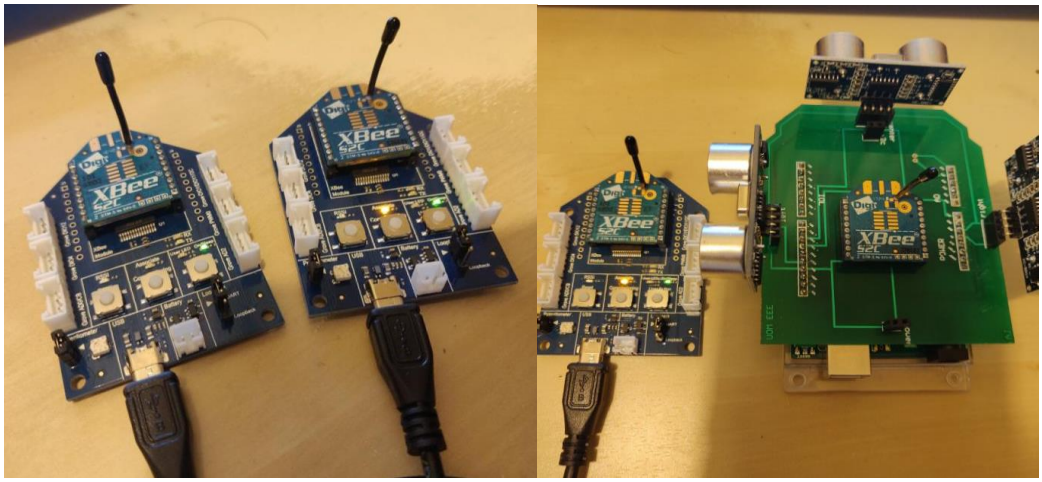


Fig26. Setup for Test A and Test B

❖ Test C: XCTU Software (AT – Mode) Point-to-MultiPoint

In Point-to-Multipoint it is necessary to develop a star network topology. i.e. broadcast mode is required by the SERVER/MASTER. This is done by setting the destination address to “00 00 00 00 00 00 FF FF” for 64-bit or “FF FF” for 16-bit.

In a similar manner to Test A, more XBee devices are configured as END/NODE. The objective is that the SERVER receives information from each node individually and is able to broadcast information throughout the network. The configuration setting for this test are shown below.

Table 4: Configuration Settings for XBee Modules in Test C

		ID	JV	DH	DL	CE	API
COORD (TX)		9999	0	0	0xFFFF	1	0
NODE 1		9999	1	0	0	0	0
NODE 2		9999	1	0	0	0	0

❖ Test D: Arduino and XCTU Software (AT – Mode) Point-to-MultiPoint

Similarly, multiple Arduino UNO boards are switched on. So that they each send their sensor status to the Server. The data received by the Server is Observed in the XCTU Software using the Console Window.

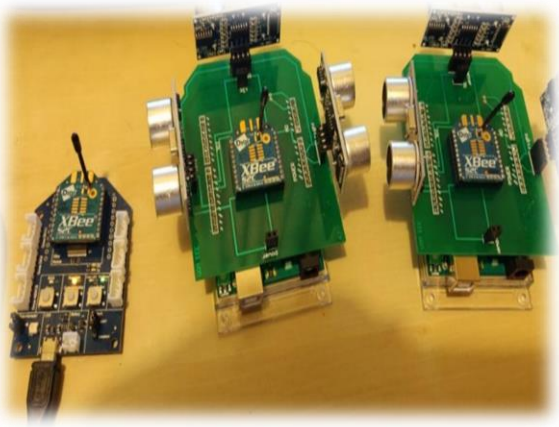


Fig27. Setup for Test D →

Result Analysis and Discussion:

Since the received message is the same transmitted message in both test A and test C. This indicated that the XBee modules were correctly configured and should work with Arduino with correct syntax.

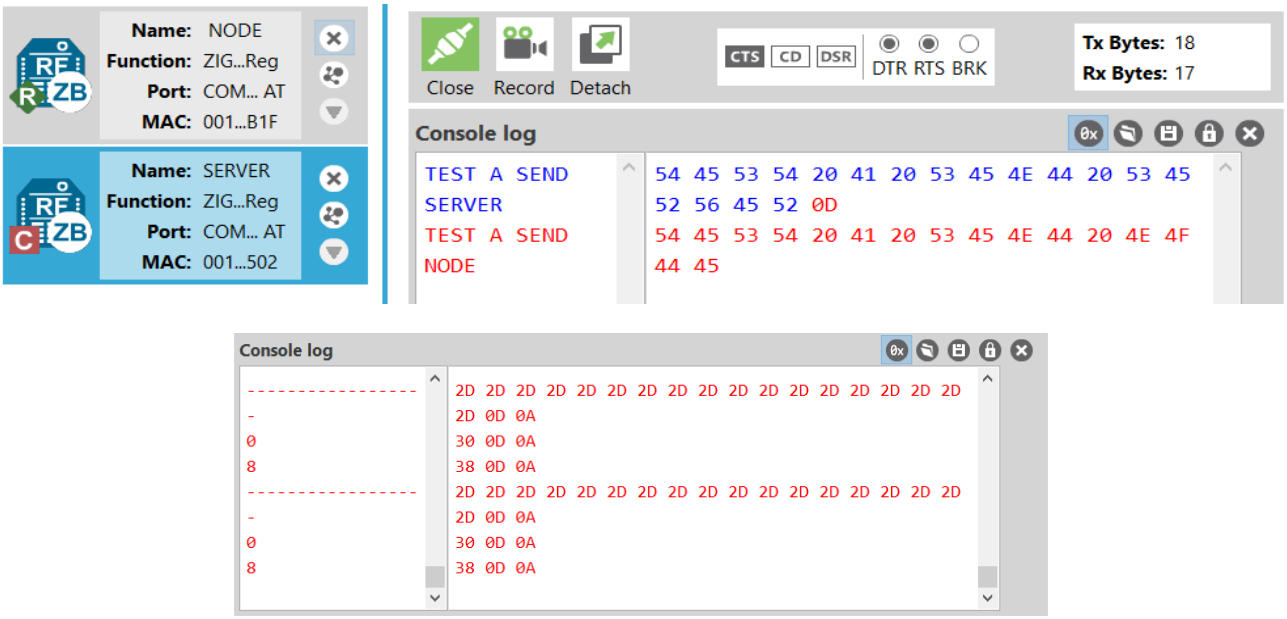


Fig28. Screenshot of Console Window for Test A and C

Figure 28 shows the results obtained from the Test A and C. The left column displays the received data in ASCII and the column on the right displays the received data in terms of hexadecimal value of each character. Blue characters represent the sent data and the red characters represent the received data. The second console window shows the data received when connected to the Arduino UNO boards 0 indicating it is in 'Mode' 0 and 8 indicating that there is no obstacle and the dashed lines used as separation between two readings.

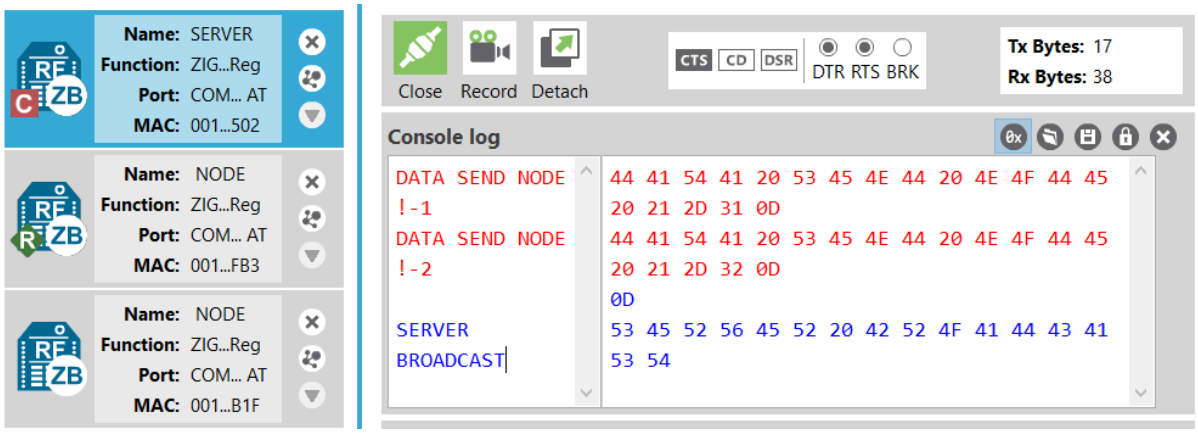


Fig29. Screenshot of Console Window for Test B

Figure 29 shows the results from Test B and D which have similar characteristics as mentioned in previous paragraph and from using the network scan using the XCTU software, all three XBee modules can be detected

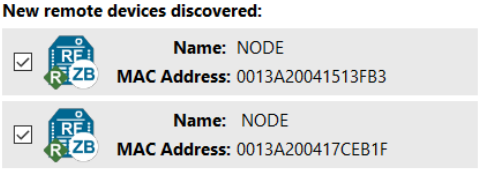


Fig30. Screenshot of Devices on the Same Network as SERVER

on the same network as seen in figure30. In broadcast mode the transmission is propagated through the entire network making setting up point-to-multipoint communication easy.

When it comes to Arduino-Arduino wireless communication using the XBee RF modules, a small problem occurs with the direct approach i.e. using Serial.read() to receive data and Serial.print() to transmit data. Since the XBee device is connected to the RX and TX port of the Arduino board it meant that the Serial port and the Transmission port were interlinked. So whenever Serial.print() or Serial.println() is used to print data on screen, it would mean that the same value is also transmitted and this would lead signal to bounce up and down while trying to read and print the value on the receiving end. This problem is solved using a feature in Arduino called SoftwareSerial. SoftwareSerial Library in Arduino allows serial communication on other digital pins of the Arduino (33). By defining a SoftwareSerial called XBee on the same ports (0, 1) and using XBee.read() to receive data and using Serial.print() to transmit data isolated the two and fixed the problem.

5.3 Transformations:

As previously mentioned, the Robocone system will tackle tricky paths and obstacles by transforming/changing into different formations whenever necessary.

In this section the various tests conducted to check changing formations and approaching final target using the Arduino UNO board, MONA and the SERVER is discussed.

5.3.1 Formations

Implementation:

Various different formations are required for different scenarios. Using the code mentioned above in Software and Theoretical Development section. Four complete robotic cone units (MONA + XBee + UNO) along with the SERVER are coded to follow a certain plan.

The objective of the plan is to start in a 2x2 formation, making sure that all the robots must start at the same time. When it detects the wall using the front ultrasonic sensors of the leaders (front two robots) it should transform into a 4x1 formation. The next step is to traverse through the narrow corridor and detect the end of the corridor so it can transform back to a 2x2 formation. All this is done with the help of synchronization of the running mode with the help of I2C and RF communication. The Mode decides which sensors stay on, which sensors stay off, which the motors run in which direction and speed.

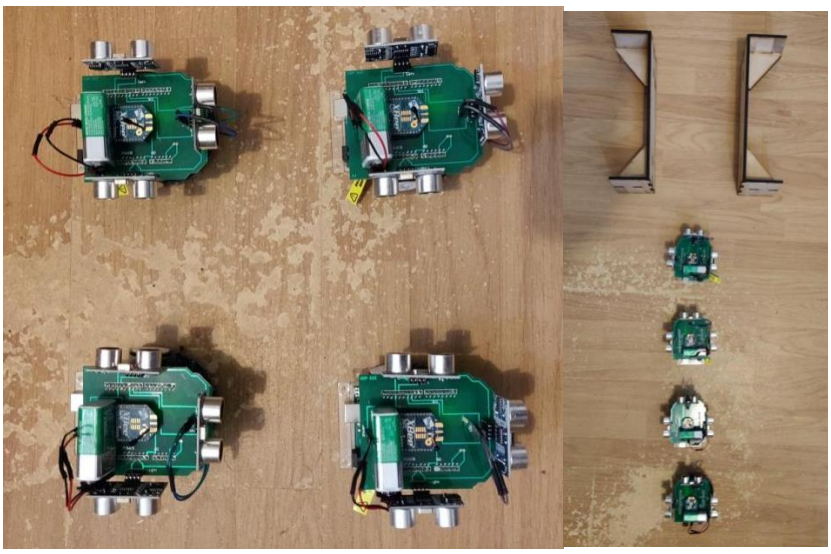


Fig31(a). Formations Test

Fig31(b). Formations Test

Result Analysis and Discussion:

First the tests were conducted individually for each robotic traffic cone unit to check if they perform their respective transformations correctly with each 'Mode' switch. Due to late delivery of parts the test for the whole system as one-unit was delayed, and only limited tests were conducted. When finally, it was tested it was noticed that the formations changes are not always consisted in respect to each other but accurate to some extent.

This is mainly because there is always some error in the PID speed control algorithm of the MONA robot. After conducting more tests, the error is found to be 1cm for every distance when travelling in a straight line and -7.5 deg every 90-degree rotation. After error compensating the code the results were more accurate but still not perfect most of the time.

The biggest challenge in the coding all of the nodes separately was matching the delays of each transformation and 'Mode' switches. It had to made sure each node has finished exiting its

loop before switching to the next ‘Mode’. To do this the approximate time to move and rotate had to be calculated and it was found to be approximately 3 seconds for moving 15cm.

Also, in ‘Mode 1’ in which the node is programmed to move in a straight line until the ‘Mode’ is switched, it is noticed that the robotic nodes move in small burst of movements. This is because it is coded to move a small distance every time it loops and because it is in an infinite loop it should move continuous however this is not the case, this is because there is some delay in updating the ‘Mode’ in each loop whenever the MONA robot receives a value for ‘Mode’ through the wired connection using I2C.

5.3.2 Final-Target Approach

Implementation:

After the Robocone system has passed through the tricky path, i.e. a narrow pathway in this case. It needs to transform into a 4x1 formation in which the system is lined up approximately parallel to the target area. After this has been done each robotic-traffic cone unit will individually approach the target until it is close enough. This is done with the help of the ultrasonic sensors. As soon as it all the nodes are close enough, they come to a stop and the mission is complete.

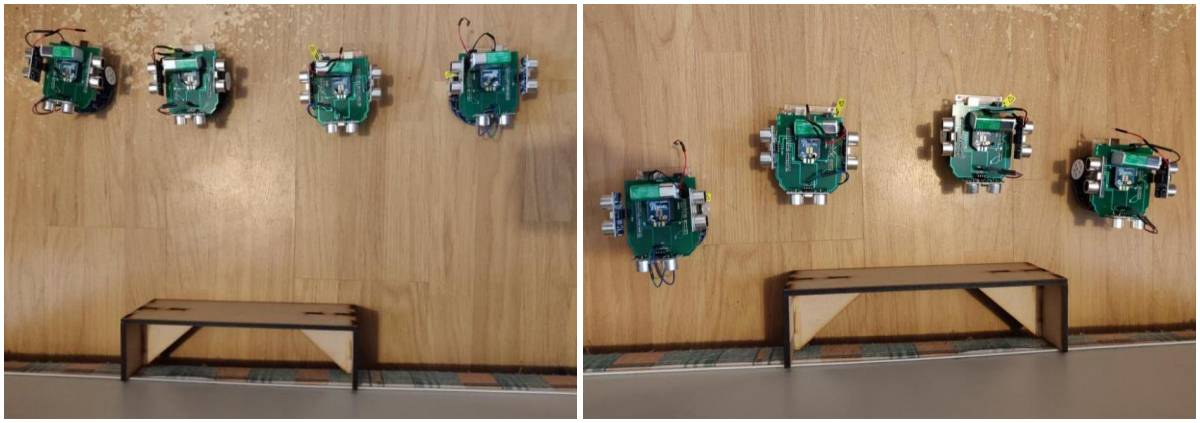


Fig32. Final-Target Approach Test

Result Analysis and Discussion:

The final part of the test was successful as the concept was simple. The result was as expected as each node when switched to the final ‘Mode’ worked individually and powered the Motor until it detected an obstacle within a certain threshold. The Main problem faced here was after all the transformations in the formations tests, due to inaccuracies in the system. The system

would always be in a straight line and would in at an angle or slightly forward/backwards. This can be solved by simply using better speed control and implementing odometry.

6. Analysis and Conclusions

6.1. Aims and Objectives

The Aims and Objectives mentioned in section 1 have all been gone through and met. MATLAB simulations were conducted, transformation plan was made, an interface between the sensor and the robot hardware was create, MONA robots were optimized and coded to perform the required mission, the communication system between all the units is setup and software was coded so that the system was able to perform obstacle avoidance and maneuver through tricky paths.

The communication system works perfectly as planned. However, rest of the designed system falls short in terms of accuracy. The speed control and localization capabilities of the robot is low due to lack of precise-odometry and sensing/mapping capability.

In short, a semi-autonomous deployment system for the Robocone project enabling it to exhibit swarming and self-organizational behavior was created.

6.2 Limitations of current system and future work

All the tests for this project were conducted with 4 robotic units instead of 16 as proposed earlier, this is mainly because of cost and time constraints. Also, the complete sensor array with the E-compass is not used and only 3 ultrasonic sensors are used for similar reasons.

Another Major problem is the Y-Axis sensing, with the current existing setup is it not possible to detect obstacles above and below the axis at which the sensor is located and if there is an object at these heights it will not be detected and will lead the system into a crash. There is no direct solution to this problem other than 3D sensing.

The current system is designed to be a model or to represent what the final system could look like. This is done to recognize what are the limiting factors to commercialize this project in a real-world scenario.

In this process it is recognized that navigating in a dynamic un-mapped environment is very hard and even if possible, a very slow and error prone task. Because the system is designed to work on roads with running traffic, this is a huge limiting factor.

This is why the project is continued assuming a pre-mapped environment with very few dynamic properties. This is not the case in reality as the environment is very dynamic and there exists not many available current technologies which could pre-map an area accurately enough for this system to operate.

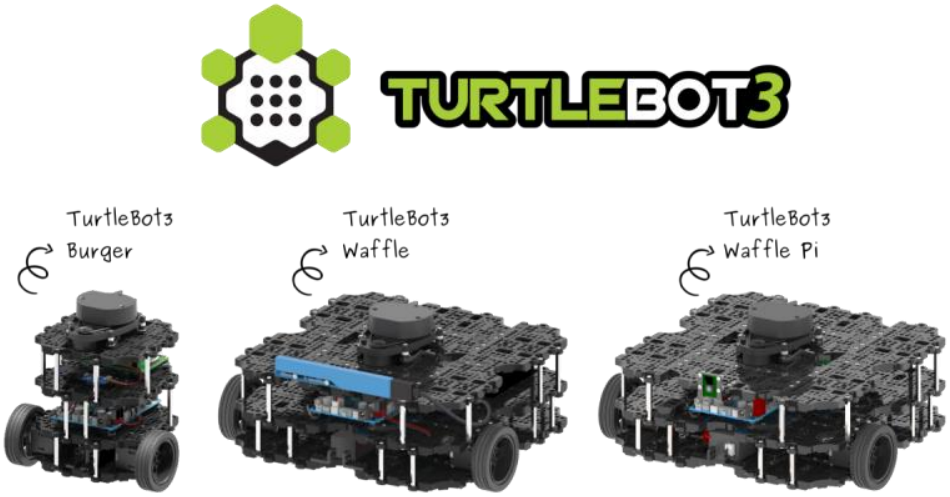


Fig33. TurtleBot (23)

The very next step toward the development of this project is too apply the same principles and idea to smarter robots than MONA. TurtleBot is perfect example. TurtleBot is a low-cost, personal robot kit with open-source software (23) which is capable of 2-D mapping an environment and navigating through it. All this is done is ROS architecture. The core technology is SLAM (simultaneous localization and mapping) algorithms to build a map and drive around it. This is perfect for the progression for this project, however there are still some limitations. The Y-Axis Sensor problem still exists in TurtleBot and the speed of operations is relatively low. Not to mention each of these robots is very fragile to work in rough/outdoor conditions and cost 1200 GBP (24). An upgrade to this is robots will be robots from Clearpath (25) which are more rugged and are even more expensive.

The main focus of future work should be dynamic mapping. A way to implement mapping algorithms such as G-mapping and Hector mapping into simple systems. G-mapping is an effective way to solve the simultaneous localization and mapping (SLAM) problem using particle filter (26) or Hector mapping which doesn't use odometry but uses the high update rate of modern LIDAR systems like UTM-30LX and provides 2D pose estimates at scan rate of the sensors (40Hz for the UTM-30LX) (27)

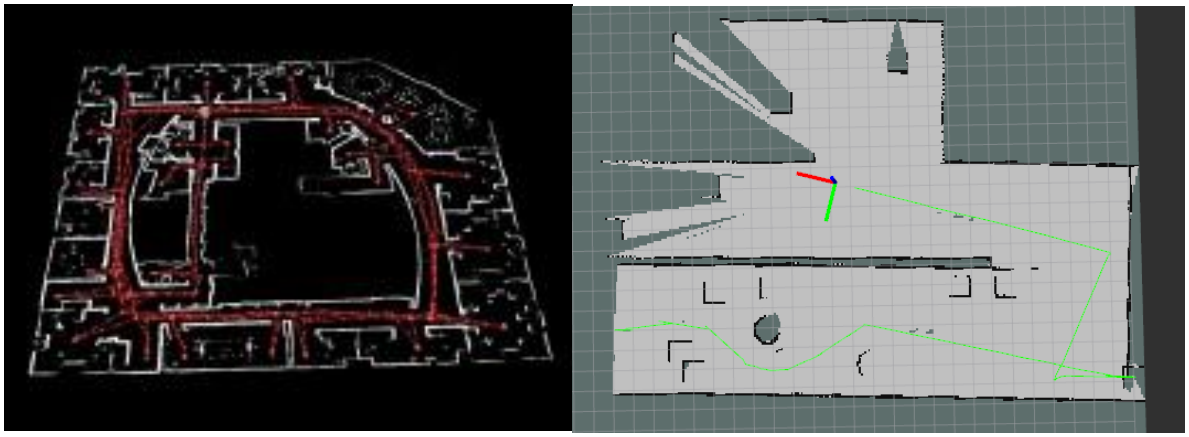


Fig34. G-mapping, left (26) and Hector Mapping, right (27)

After all this it still need to made sure that the end product economically feasible and easily manufacturable.

6.3 Conclusion

This project was a success as the final product was able to tackle the tricky pathways and cone of the decided area. The UNO Board acts as a computational board by reading the sensor status, updating the Server and then informing the MONA robot. The Server acts as a Master unit and broadcasts the update in 'Mode' whenever it receives the necessary sensor input from the Arduino UNO board and this data it transmitted to the MONA robot using the I2C interface. Also, the MONA robot is used for speed control using PID control and the encoders present on the board.

Finally, ZigBee communication platform (XBee) is the best fit for operation as the communication system in this project. It allows easy point to multipoint communication along with many more benefits over the other platforms.

The system is not ready for real world application as the working environment is very dangerous and any errors or faults can lead to serious accidents.

Despite achieving the main objectives of this project. The project is very far from completion due to loads of limitation. The concept of having autonomous robots acting as traffic cones in running traffic still remains as an achievable target. In the near future when the required technology becomes much cheaper and easier to access this system can be easily developed and will be seen on the roads saving many workers lives.

7. References

[1] Hse.gov.uk. (2018). Workplace fatal injuries in Great Britain 2018. [online] Available at: http://www.hse.gov.uk/statistics/pdf/fatalinjuries.pdf [Accessed 14 Jan. 2018]

[2] Zhong-yang, Z. and Tan, Y. (2013). Research Advance in Swarm Robotics. [online] ScienceDirect. Available at:https://www.sciencedirect.com/science/article/pii/S221491471300024X [Accessed 10 Jan. 2019].

[3] Kitts, C. and Mas, I. (2011). Centralized and decentralized multi-robot control methods using the cluster space control framework. [online] ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/abstract/document/5695768 [Accessed 20 Jan. 2019].

[4] Junaid Rehman, (2018), Types of systems [online]. Available at: http://www.itrelease.com/2017/11/difference-centralized-decentralized-distributed-processing/centralized-vs-decentralized-vs-distributed-processing/ [Accessed 16 November 2018].

[5] 'Swarm robotics' (2015) Wikipedia. Available at https://en.wikipedia.org/wiki/Swarm_robotics [Accessed: 25 October,2018].

[6] Reynolds, C. (2001). Boids-Background and Update. [online] Available at: http://www.red3d.com/cwr/boids/ [Accessed 7 Dec. 2018].

[7] 'Swarm behaviour' (2015) Wikipedia. Available at https://en.wikipedia.org/wiki/Swarm_behaviour#Mathematical_models [Accessed: 26 October,2018]

[8] Bulc, V. (2018). Road Safety in the European Union – Trends, statistics and main challenges. [online] ec.europa.eu. Available at: https://ec.europa.eu/transport/road_safety/sites/roadsafety/files/vademecum_2018.pdf [Accessed 13 Jan. 2019].

[9] Fredslund, J. and Matari´, M. (n.d.). A General Algorithm for Robot Formations Using Local Sensing and Minimal Communication.´

[10] Hui Wen Lim, Robocone: Robocone Traffic Cone Third Year Individual Project : Final Report (2018) [Accessed 25 Oct. 2018]

[11] DIGI, XBee®Zigbee®Mesh Kit RadioFrequency(RF) Module (2018) Digi, "Wireless Communication", Digi International Inc. [online] Available at:

https://www.digi.com/resources/documentation/Digidocs/90001456-13/containers/cont_wireless_comm.html [Accessed 6 Jan. 2019]

[12] Digi, "RF Kits Common", Digi International Inc. [online] Available at: http://docs.digi.com/display/RFKitsCommon/RF+Kits+Common [Accessed 23 Feb 2019]

[13] Digi International Inc. (2018). Comparison of transparent and API modes. [online] Available at: https://www.digi.com/resources/documentation/Digidocs/90001942-13/concepts/c_xbee_comparing_at_api_modes.htm?TocPath=How%20XBee%20devices%20work%7CSerial%20communication%7C_____2 [Accessed 6 Jan. 2019].

[14] MonaRobot, "Mona-Platform", (2017), GitHub repository. [online] Available at: https://github.com/MonaRobot/Mona-Platform [Accessed 24 November 2018]

[15] F.Arvin, S.Watson, A.E.Turgut, J. Espinosa, T.Krajník, B.Lennox "Perpetual Robot Swarm: Long-Term Autonomy of Mobile Robots Using On-the-fly Inductive Charging", Journal of Intelligent & Robotic Systems, (2017). [online] Available at: https://www.researchgate.net/publication/320292786_Perpetual_Robot_Swar m_Long-Term_Autonomy_of_Mobile_Robots_Using_On-thefly_Inductive_Charging [Accessed 26 November,2018]

[16] joekeo, "MONA", (2017), GitHub repository. [online] Available at: https://github.com/joekeo/MONA [Accessed 24 November,2018]

[17] Farshad Arvin, “Mona Robot Project” (2017). [online] Available at: http://www.monarobot.uk/ [Accessed 26 November,2018]

[18] MonaRobot, "Mona Robot", Arduino Project hub. [online] Available at: https://create.arduino.cc/projecthub/university-of-manchester-roboticlab/mona-robot-ba39d2 [Accessed 24 November,2018]

[19] Electro Schematics. (n.d.). HC-SR04 Datasheet. [online] Available at: https://www.electroschematics.com/8902/hc-sr04-datasheet/ [Accessed 2 Apr. 2019].

[20] Random Nerd Tutorials. (n.d.). Complete Guide for Ultrasonic Sensor HC-SR04 with Arduino. [online] Available at: https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/ [Accessed 3 Feb. 2019].

[21] Instructables. (n.d.). I2C Between Arduinos. [online] Available at: 21. https://www.instructables.com/id/I2C-between-Arduinos/ [Accessed 6 Mar. 2019].

[22] Ackerman, E. (2013). TurtleBot Inventors Tell Us Everything About the Robot. [online] IEEE Spectrum. Available at: https://spectrum.ieee.org/automaton/robotics/diy/interview-turtlebot-inventors-tell-us-everything-about-the-robot [Accessed 25 Mar. 2019].

[23] TurtleBot. (2014). TurtleBot. [online] Available at: https://www.turtlebot.com/about/ [Accessed 27 Mar. 2019].

[24] TurtleBot3 Waffle Pi. TurtleBot. [online] Available at: https://robosavvy.com/store/turtlebot3-waffle-pi.html?___store=eng [Accessed 27 Mar. 2019].

[25] ClearPath robotics. [online] Available at: https://www.clearpathrobotics.com/robotic-automation-services/

[26] Grisetti, G, Stachniss. G, Burgard.W (2018). G Mapping. [online] Open SLAM. Available at: https://openslam-org.github.io/gmapping.html [Accessed 26 Mar. 2019].

[27] Kohlbrecher, S. (2012). hector_mapping. [online] http://wiki.ros.org. Available at: http://wiki.ros.org/hector_mapping [Accessed 26 Mar. 2019].

[28] Jonathan A. Titus, “The Hands-on XBee Lab Manual: Experiments that Teach you XBee Wireless Communications”, (2013). [online] Available at: https://www.sciencedirect.com/science/book/9780123914040 [Accessed 26 Feb. 2018].

[29] Matt Hillman, "An Overview of ZigBee Networks: A guide for implementers and security testers", MWR InfoSecurity. [online] Available at: https://www.mwrinfosecurity.com/assets/Whitepapers/mwri-zigbee-overviewfinalv2.pdf [Accessed 2 Feb. 2018].

[30] Simonsen, S. (2018). How Swarm Intelligence Is Making Simple Tech Much Smarter. [online] Singularity University. Available at: https://singularityhub.com/2018/02/08/how-swarm-intelligence-is-making-simple-tech-much-smarter/#sm.0000d8r7g94e8enfr86236v82arxj [Accessed 13 Mar. 2019].

[31] Clarence W de Silva, "Some Issues and Applications of Multi-robot Cooperation", IEEE, (2016). [online] Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7565951 [Accessed 29 Feb 2019]

[32] Belkacem Khaldi, Foudil Cherif, “An Overview of Swarm Robotics: Swarm Intelligence Applied to Multi-robotics”, University of Biskra, (2015). [online] Available at: https://pdfs.semanticscholar.org/e545/ed969935295cd1fc2c45ae43aeaf3fd9e 45e.pdf [Accessed 28 April 2018].

[33] Arduino.cc. (2018). *SoftwareSerial Library*. [online] Available at: https://www.arduino.cc/en/Reference/SoftwareSerial [Accessed 16 Mar. 2019].

8. Appendices

8.1 PROGRESS REPORT

ROBOCONE: Robotic Traffic Cone

Swarming algorithms – Rule-based Autonomous Deployment

Third Year Individual Project – Draft Introduction

Abstract

Robocone is a system of a group of robotic traffic cones which will replace the traditional traffic cones used today. This project focuses on autonomous deployment of Robocone, meaning the robot-nodes should be capable of navigating the pre-mapped environment though measuring and sensing other robot-nodes and obstacle and changing formation accordingly. In this report, it is assumed that 16 robot nodes are going to be used and the transformation into different formations is done accordingly. Using MATLAB as a simulation tool by plotting the micro-movements in every transformation, the most efficient and fastest way of transformation is selected. Multiple MONAs that represent the robot-nodes in this project, co-ordinate with a master for a master- slave system. Based on the previously conducted research on this project, a number of tests have been done to support the implementation of the project. Via the Arduino board on the MONA robots it is instructed to perform movement and communicate between robot-nodes. To encourage swarm behaviour, sensors and radio modules are used to communicate with other nodes and position each node locally. A range of test is conducted to find the best solution for the sensing of other robots and obstacles. The MONA's are coded in Arduino such that they are able to perform the transformation algorithm whilst traversing through a pre-mapped environment.

Contents

a. Aims and Objectives
b. Limitation of current system and future work
c. Conclusion

7 References

8 Appendices

a. Appendix 1 – Proposal Report
b. Appendix 2 – MATLAB Simulation Code
c. Appendix 3 – MONA Arduino Code

1.Introduction

1.1 Introduction

This report documents a project undertaken in which the autonomous deployment of robotic traffic cone is investigated, which is also referred to as the 'Robocone'. The most efficient and practical ways to carry out the deployment will be researched, exploring the challenges and limitations. Transformation into different formation will be looked into and algorithms for doing so and dealing with the obstacles and path detection will we decided. Simulations will be conducted in order to do so. Following this, sensing solution will be decided with the necessary research and tests commenced. The results will then be used for implementation in robotic-nodes coded to demonstrate navigation through a pre-mapped environment.

1.2 Motivation

For road maintenance work, we use a vast number of traffic cones to redirect traffic to ensure the safety of drivers and so that the maintenance work can be done efficiently. However, the deployment of these cones has been done manually; this not only takes up loads of time and manpower but also puts the workers life at risk.

If this could be done autonomously, it would save loads of time and labour also ensuring that the job is done with great precision therefore decreasing the chances of any man-made errors.

Certain complex natural swarming behaviour in nature inspire us to implement similar type of systems for our robotic traffic cone system, so that each module in the system can interact with the environment and each other, especially for a group of robots that have to demonstrate self-organising capabilities and cooperation within the system.

Therefore, it would be useful to implement an autonomous deployment system which can be remotely controlled through a wireless controller allowing it communicated within a network so navigate through a pre-mapped environment avoiding the obstacles in doing so.

1.3 Aim

The scope of this project is to develop a semi-autonomous system for the robocone project enabling it to exhibit swarming behaviour. Allowing multiple robot units to set up in a given formation in the required location that is set by a controller and the robots would also be expected to manoeuvre around tricky environment, avoiding obstacles and changing formation to get through small areas.

A viable robot will be selected and then additional sensors along with computations boards will be chosen based on research and simulations. The robot-nodes will be coded in respective

software alongside with the sensor to demonstrate the movement of robocone system in a respective environment.

1.4 Objectives

Whilst trying to achieve the aims that has been discussed above, the main would be objectives would be:

- To be able to code in MATLAB to conduct simulation of transformation models.
- To be able to program with Arduino to code the MONA robot-nodes.
- To learn the algorithm of robots to perform swarm and interaction.
- Code software for the robot-nodes to show movement and perform transformation when required.
- Establish which sensors are to be used.
- Create an interface between the chosen sensor and the robot hardware.
- To implement a wireless communication system between a server and multiple nodes.
- Optimise software so that the robot-nodes are able to avoid obstacles by being environment aware with the help of the sensors and also being able to traverse through tricky paths.

7. Background research and literature review

2.1 Swarm Robotics

Swarm robotics is defined as a new approach to the co-ordination of multiple robotic systems consisting of large number of relatively simple, small and low-cost robots in terms of their physical body and their controlling behaviour which take advantage of a large population (1). A key component of the system is the local communication between the group enabling it to interact with other robots of the group and also the immediate environment.

2.1.1 Swarm robotics takes advantage of being a de-centralized and distributed system.

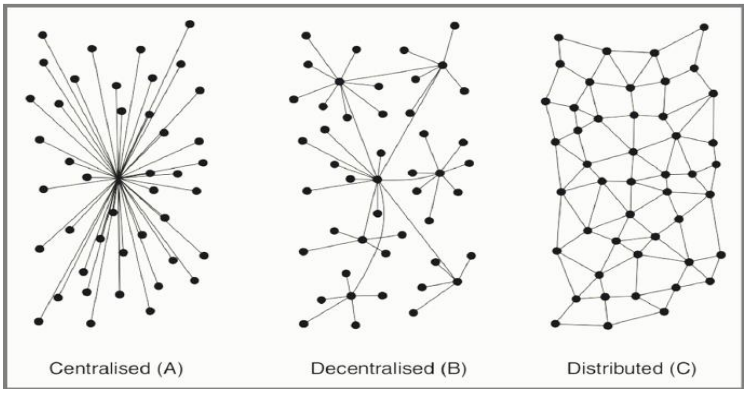


Fig1. Types of Systems (2)

Centralized systems mean one controller commands everything although it is simple, easy to make and easy to control, if the core controller is damaged or isn't available all the subsystem fails and also the central controller is expected to be fast in making decisions from the

information it receives from the slave sensors and provide commands to the actuators which means it is easy to code but quickly gets complicated and expensive as the scaling increases.

Decentralized system solves both the problems by having multiple controllers and if failure is encountered in one of the controllers the other parts of the robot are still working. This facilitates easy scalability and addition of new features. It takes longer to design and code this type of system to allow for abstraction and scalability.

2.1.2 Advantages of swarm robotics:

- Compared to a single robot, to complete a sophisticated task, the robot must be designed with complicated structure and control modules resulting in high cost of design, construction and maintenance. Also a single robot might be vulnerable especially when a broken part may affect the whole robot and it is difficult to debug in such situations. A swarm robotic group will achieve the same through inter group cooperation.
- Parallelism: The size of a swarm robotic group is usually large and it deals with multiple targets at once enabling it to perform tasks that are distributed in a vast environment, saving time.
- Scalable: Since a swarm group is based on a non-centralized system, it allows the individuals to join or quit the task at any time without interrupting the swarm as a whole. This also tells us that the system is adaptable for different sizes of population with minor modifications to the software and hardware.
- Economical: As discussed before the cost of swarm robotics is significantly low and as a whole cheaper that a complex single robot because the individual robotic nodes are mass produced and a single complex system requires precision machining.
- Energy efficient: The individuals in a swarm are smile and small robots which have much lower power consumption when being compared to a single robot. This also means that in most cases they have more battery life meaning they are better in wireless situations.

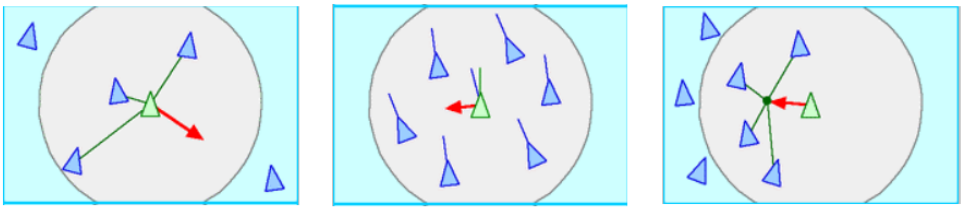
2.2 Models of Swarm robots

A plain set of rules at individual level can produce a large set of complex behaviours at the swarm level.

One of the simplest models to demonstrate swarm behaviour is Boids computer program (4) which works on three principles based on dimensional computational geometry.

These were:

- Separation: steer to avoid crowding local group
- Alignment: steer towards average heading of local group
- Cohesion: steer to move toward the position of local group



Seperation Allignment Cohesion

Fig 2. Boids swarming model

These three rules made sure the group always maintained a minimum and maximum distance between each other while being centred and group whilst moving towards a target.

A slightly more detailed model is discussed by Fredslund solves localisation of robotic nodes by simple referencing methods. In this model each robotic node has sensing capability and a unique identifier and a radio communication model. The ID is broadcasted regularly so each robot knows how many robots are participating in the formation and each robot has its reference. By Referencing and position itself according to its partner/reference each robot can be localized and transformation can occur with simple commands.

For this project sensors will be selected to perform obstacle avoidance and also will be used for localisation which will be discussed later after testing and debugging has occurred.

2.3 Transformation models

For the proposed system to work i.e. the robotic nodes to cone off a certain pre-determined are of the road or construction site. The Robotic system will have to follow certain formations.

It is decided that for research and development process sixteen robot-nodes will be used to represent the system however this system will easily be scalable up to whatever number of robotics cones are required for the operation.

These 16 robotic nodes will have to be deployed off a vehicle or a station. A 4x4 square formation is decided as the best option for the initial deployment stage. It is the most compact formation where the nodes maintain a minimum distance from each other, this formation also enables the robotic nodes to be charged wirelessly through inductive charging in a charging station which is represented in the figure below.

Fig 3. Wireless Charing Representation

These 16 robotic nodes will have to undergo various transformations to maneuver challenging pathways with the basic transformation being changing to a double file formation and then two a single file formation. Considering maximum area required and the number of steps required to carry out the transformation it was decided that this will be done by interlocking the inner columns with the outer columns.

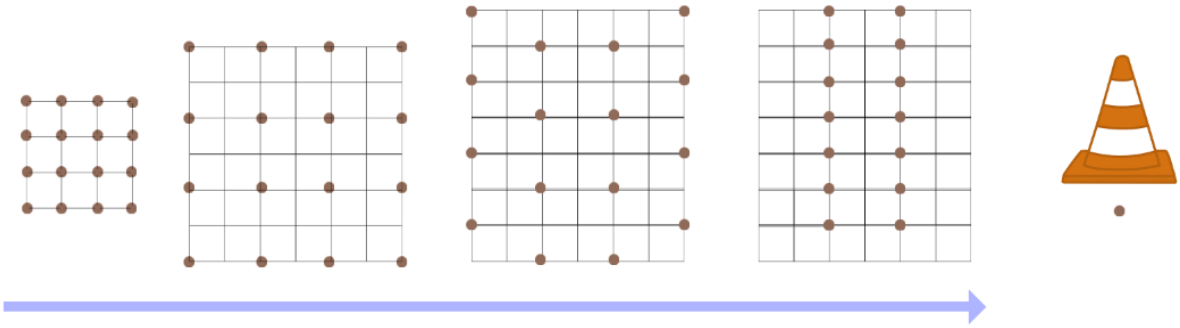
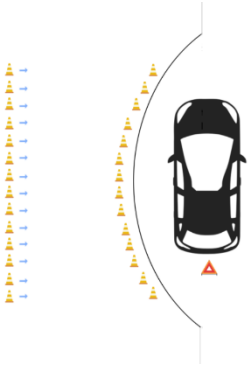


Fig 4. Transformation to a file of two

To achieve this, the grid expands and then the first and last columns move upwards and move towards the center merging into a file of two as shown in the figure. Similarly, this file of two will transform to a file of one by interlocking of the two columns.

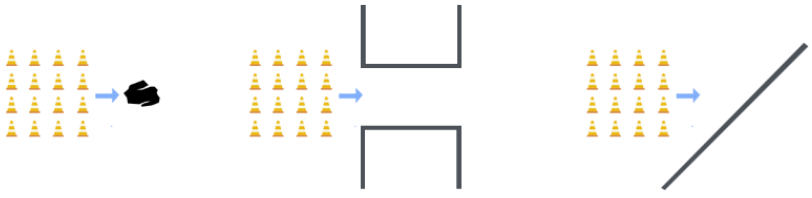


The final stage of every mission will be to enclose the designated area. This will be done with the help of the sensors available on the robots. The robots will move towards the target once they are in a single file formation placed perpendicular to the target. With proximity sensors and radio modules to communicated with other members of the group the robots will maintain a fixed distance and a maximum angle with each other while trying to achieve a minimum distance with the target.

This model will later be discussed in the simulation section where the different transformations will be demonstrated with the help of MATLAB's plotting tools.

2.4 Environment Analysis

This project assumes that the roads are pre-mapped with respective hardware and software, also that the mission is pre-determined. However, roads are not always obstacle free and the paths are tricky. Robocone needs to deal with these kinds of situations and avoid stray obstacles, some examples are shown below. Robocone should be able to make smart transformations according to the path which will be detected by one of the robotic node's sensors and communicated to the rest of the swarm to carry out the transformation to the respective formation.



2.5 Wireless Communication Model

For the purpose of this project we will need one server communicating with multiple Robocone nodes wirelessly. The server receives sensor information from different nodes computes it to decide which working mode it should operate in and informs all the nodes to change its working mode if necessary. This can be done using Bluetooth or Zigbee architecture. In the next section we will discuss which one is the most suitable of this operation.

2.5.1 Comparison between Zigbee and Bluetooth:

The two most common wireless personal area network (WPAN) standards established by the institute of Electrical and Electronic Engineers (IEEE) under IEEE 802.15 are Bluetooth (IEEE 802.15.1) and ZigBee (IEEE 802.15.4).

Shown below is a spec comparison between Bluetooth and Zigbee. Zigbee architecture protocol is selected as it is a more suitable option to generate a good communication network after comparing factors such as range, length of messages, propagation time, area and interference.

	Bluetooth (IEEE 802.15.1)	ZigBee (IEEE 802.15.4)
Representative	HC-05 (Bluetooth 2.0)	XB24-AWI-001 (XBee Series 1)
Range	10m	30m
Networking Topology	Ad-hoc, Piconet, Scatter net	Ad-hoc, P2P, Star, Mesh
Operating Frequency	2.4 GHz	868 MHz (EU), 915 MHz (US & Aus), 2.4 GHz (worldwide)
Power Consumption	Supply Voltage: 3.3V-3.6VDC Working Current: <50mA Transmit Power: +4dBm	Supply Voltage: 2.8-3.4VDC Transmit Current: 45mA@3.3VDC Receive Current: 50mA@3.3VDC Transmit Power: 1mW(+0dBm)
Modulation Technique Spreading	Frequency Hopping Spread Spectrum (FHSS)	Direct Sequence Spread Spectrum (DSSS)
MAC Scheme	FH-CDMA	CSMA-CA
Protocol Stack Size	250 Kbyte	28 Kbyte
Data Rate	1 Mbit/s	250 Kbit/s (2.4GHz band)
Latency	3-10s	30 ms
Number of nodes per master	7	64K
Cost	Low (Up to £7)	High (Up to £20)

The Zigbee architecture can perform broadcasting and allow scalability to perform star network topology that is required in this project. Since Bluetooth takes longer to perform point-to-multipoint

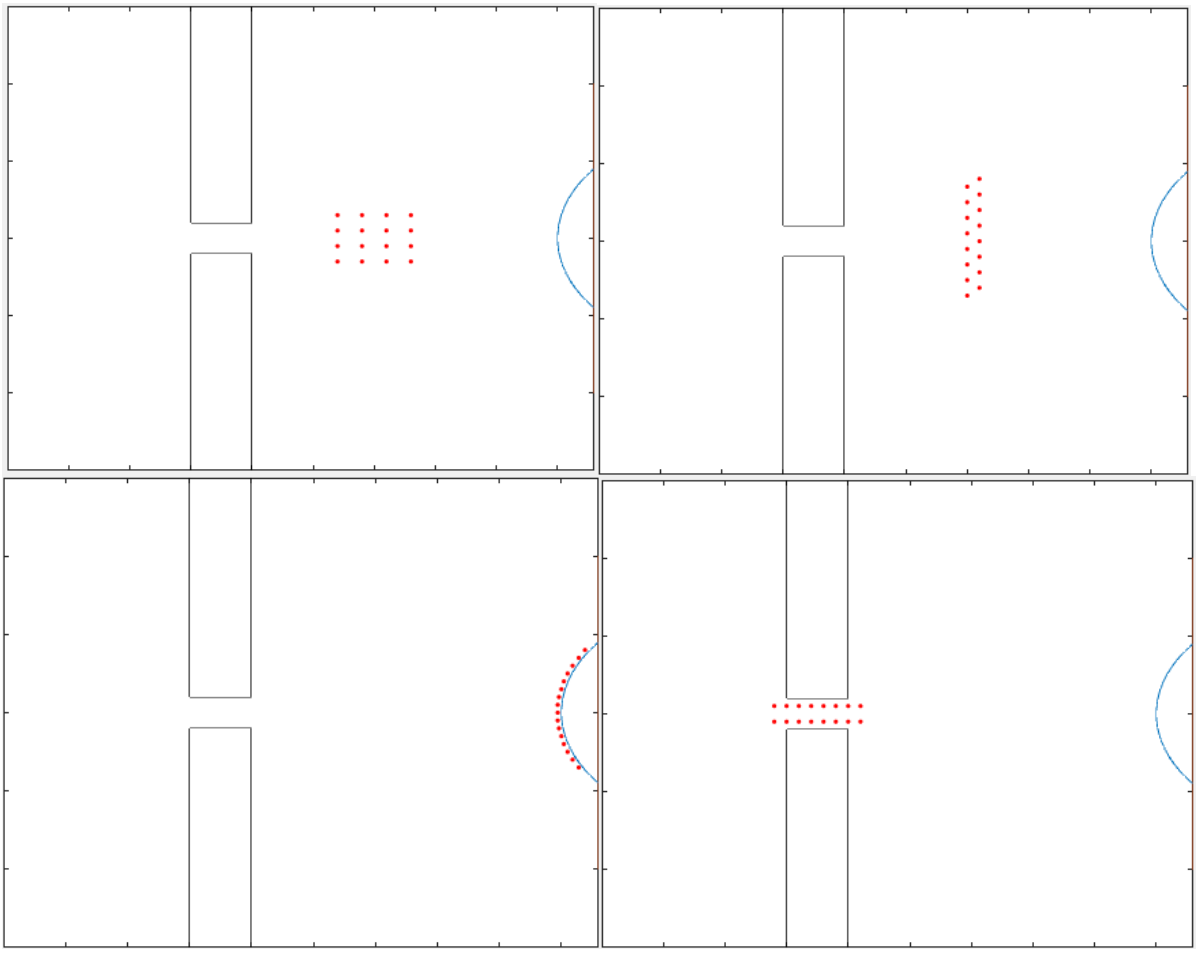
communication through piconet as master only connects to one slave device at a time Zigbee is the obvious choice here. Zigbee also works with low data rate, has low power consumption and has intermittent data transmissions for control purposes.

Zigbee also supports immediate join time and does not require unobstructed line-of-sight and less complexity as compared to Bluetooth.

MATLAB Simulations

3.1 MATLAB

A program in MATLAB was created to demonstrate the transformation models to perform obstacle avoidance and final mission. The screenshots below show a few stages of the MATLAB simulation.

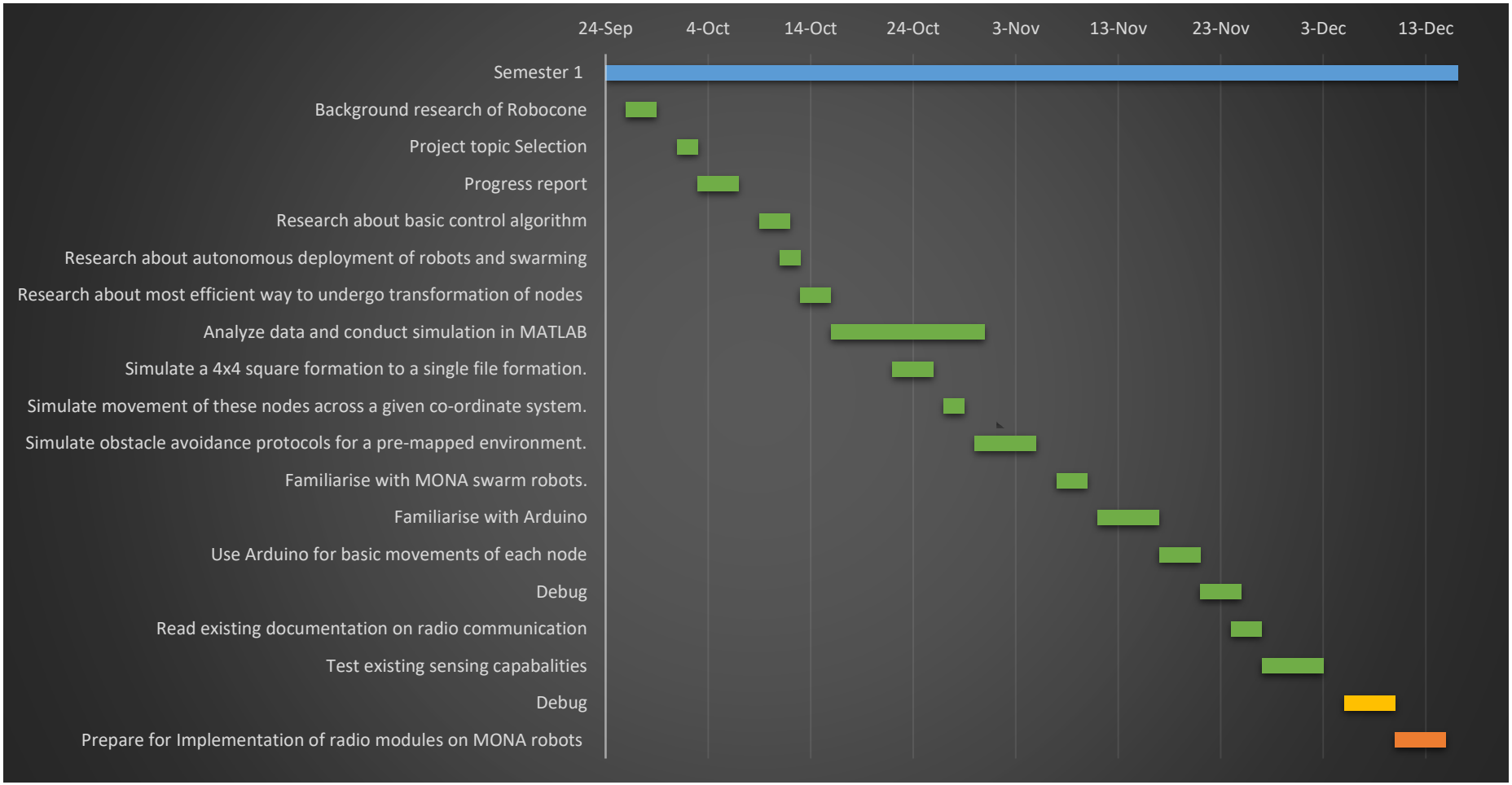


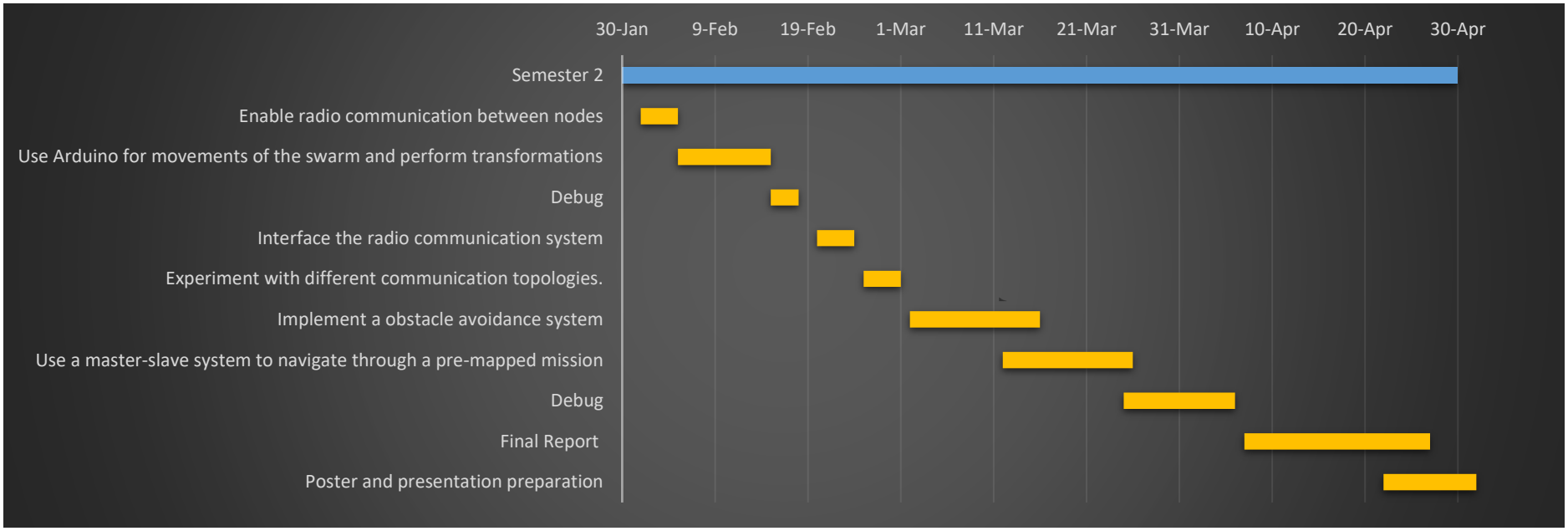
References (rough work):

1. https://www.sciencedirect.com/science/article/pii/S221491471300024X
2. Junaid Rehman, (2018), *Types of systems* [ONLINE]. Available at: http://www.itrelease.com/2017/11/difference-centralized-decentralized-distributed-processing/centralized-vs-decentralized-vs-distributed-processing/ [Accessed 16 November 2018].
3. https://en.wikipedia.org/wiki/Swarm_behaviour#Mathematical_models
4. http://www.red3d.com/cwr/boids/

8.2 Project Plan

The Project Plan below shows the task that have been completed before Christmas break and future plans until submission is mentioned. (Green → finished / Yellow → due). This Project plan was designed at the end of Semester 1.





The plan for Semester two was mostly followed until 20th of March. There was a slight delay in progress after this date due to late delivery of parts

8.3 General Risk Assessment Form

Date:	Assessed by:	Checked / Validated* by:	Location:	Assessment ref no	Review date:
10/10/2018	Ajaykumar Selvam Mudaliar	Dr. Peter N Green	SSB C34 Lab, Barnes Wallis PC Cluster		3/2/2018

Task / premises:

Soldering, Board construction, wire connections, Use of laboratory equipment such as DMM, oscilloscope and other test equipment. Programming using PC, keyboard, mouse and other necessary hardware connections between Arduino, MONA, PCB and RF modules.

Activity	Hazard	Who might be harmed and how	Existing measures to control risk	Risk rating	Result
Soldering PCB	Burns from fire	Any individual with access to the room or building.	• Soldering work must be performed on a heat-proof mat with no flammable materials close-by.	Medium	A
Soldering PCB	Burns from soldering iron respiratory issues from solder fume	Any individual with access to the room or building.	• Fumes extractors must be used while soldering. • Secured soldering iron holders are provided for soldering work. • Soldering work must be carried out with care.	Low	T
Soldering PCB	Eye damage from lead trimming or solder splashes.	Any individual with access to the room or building.	• Do not point the end of the solder iron to be trimmed directly towards eye or other sensitive parts. • Excess solder must be removed from the tip of solder iron with wet sponge. • Remove excess solder from the board using a solder pump.	Low	T

Activity	Hazard	Who might be harmed and how	Existing measures to control risk	Risk rating	Result
Working in the laboratory or PC cluster	Electrical shocks	Any individual with access to the room or building.	• Equipment is tested and inspected at regular intervals to insure safety. • Faults and damage must be reported to respective staff. • Protective equipment must be used if required.	Medium	A
Working in the laboratory or PC Cluster	Repetitive strain injury (RSI), headache or eyestrain from regular computer use	Any individual with access to the room or building.	• Breaks between working hours to ensure rest time. • Proper seating, lighting and desks should be provided. • Display Screen Equipment (DSE) assessment undertaken.	Low	T
Working with Mona Robots.	Sitting/Stepping on the robots	Any individual with access to the room or building.	• Storing the robots in a fixed place after testing/use to ensure safety.	Low	T
Board construction and wire connection	Cuts/abrasions	Any individual with access to the room or building.	• Demonstration and moderate supervision required to handle unfamiliar tools or more dangerous equipment. • Wear protective clothing (gloves, goggles etc)	Low	T
Hardware connection	Exploding Electrolytic capacitors	Any individual with access to the room or building.	• Make sure the polarity of all electrolytic capacitors are correct before powering circuit.	Medium	T

Activity	Hazard	Who might be harmed and how	Existing measures to control risk	Risk rating	Result
Hardware connection (Installing or removing components)	Damage to the component	Any individual with access to the room or building.	• Handle the Mona robots with care when assembling to avoid damage to the robot.	Low	T
Going to Lab/PC-Clusters	Injuries from slipping/tripping	Any individual with access to the room or building.	• Working area should be kept clear of any obstructions. • Ensure enough lighting in the room or building. • Make sure cables are secured and kept tidy after use. • Any spillages or damage in the area should be reported and dealt with immediately.	Low	T

Risk Rating: LOW - most unlikely that harm would arise under controlled conditions listed;

MEDIUM - more likely that harm might occur, and the outcome could be more serious

HIGH - if injury is likely to arise, that might be serious or even a fatality

Result: T = trivial risk ; A = adequately controlled, no further action necessary; N = not adequately controlled, actions

8.4 CODE

A: MONA MAIN CODE FOR ONE OF THE FOUR ROBOTS

```
//#include <SPI.h>
#include <Wire.h>
char Mode = '0';
#include <MsTimer2.h>
int LED1 = 13;       //D13 RED LED on top
int LED2 = 12;       //D12 bottom front LED
#define ON   1
#define OFF  0
//motors IOs  ---------------------------
int Motor_right_PWM = 10;  //   0 (min speed) - 255 (max speed)
int Motor_right_direction = 5;  //   0 Forward - 1 Reverse
int Motor_left_PWM = 9;    //   0 (min speed) - 255 (max speed)
int Motor_left_direction = 6;   //   0 Forward - 1 Reverse
#define Forward 0            // Direction code
#define Reverse 1            // Direction code
int directl=0;
int directr=0;
int directll=1;
int directrr=1;
int directlreal=1;
int directrreal=1;
const byte Left_interruptPin = 2;
const byte Right_interruptPin = 3;

int Left_forward_speed=0;
int Right_forward_speed=0;
float Kp = 0.15;
float Ki = 3;

// Used to control whether this node is sending or receiving
```

```
// Used to control whether this node is sending or receiving


//------------------------------------------------------
void forward(){
  analogWrite(Motor_right_PWM,Right_forward_speed); // right motor
  digitalWrite(Motor_right_direction,directr); //right
  analogWrite(Motor_left_PWM,Left_forward_speed); // left motor
  digitalWrite(Motor_left_direction,directl); //left
}

long int Left_counter, Right_counter;

//------------------------------------------------------
void Left_int_counter(){
  Left_counter++;
}
//------------------------------------------------------
void Right_int_counter(){
  Right_counter++;
}
//------------------------------------------------------

float Left_compensatory=1,Right_compensatory=1;
int Desiredr = 0;   // number of pulses which is desired
int Desiredl = 0;   // number of pulses which is desired
int run_exp=0;

float sum_Right_compensatory=0,sum_Left_compensatory=0;

//------------------------------------------------------
void Timer_overflow() {   // every 400 ms is called
  run_exp++;
  Right_compensatory =  directrr*Desiredr - directrreal* Right_counter;
```

```
 Left_compensatory  =  directll*Desiredl -  directlreal* Left_counter;
 sum_Right_compensatory +=  Right_compensatory;
 sum_Left_compensatory += Left_compensatory;

 // a basic proportional control (disable these two lines for Open-loop control)
 //Right_forward_speed +=  Kp * Right_compensatory;
 //Left_forward_speed  +=  Kp *  Left_compensatory;
 Right_forward_speed =  Kp * Right_compensatory+Ki*sum_Right_compensatory*0.05;
 Left_forward_speed  =  Kp * Left_compensatory+Ki*sum_Left_compensatory*0.05;

 if ( Right_forward_speed<0){
// Right_forward_speed=abs(Right_forward_speed);
  directrreal=-1;
 directr=1;
 }
 else{
  directrreal=1;
  directr=0;
 }

 if ( Left_forward_speed<0){
 // Left_forward_speed=abs(Left_forward_speed);
  directlreal=-1;
  directl=1;
 }
 else{
  directlreal=1;
  directl=0;
 }
 forward();  // update PWM set-point
 //reset counters
 Left_counter=0;
 Right_counter=0;
}
```

```
void setup() {
  Wire.begin(9);                // join i2c bus with address #8
  Wire.onReceive(receiveEvent); // register event
  Serial.begin(9600);           // start serial for output
  // set encoder counter to 0
  Left_counter=0;
  Right_counter=0;
  // init INT0 and INT1 for left and right motors encoders
  pinMode(Left_interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(Left_interruptPin), Left_int_counter, CHANGE);
  pinMode(Right_interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(Right_interruptPin), Right_int_counter, CHANGE);
  // setup Timer2 for motor control
  MsTimer2::set(50, Timer_overflow); // 400ms period
  MsTimer2::start();
  //set PWM set-point
  forward();
}
//-----------------------------------------------------
//-----------------------------------------------------
void receiveEvent(int bytes) {
  Mode = Wire.read();           // receive byte as an integer
  Serial.print(Mode);
}


//-----------------------------------------------------
//-----------------------------------------------------
void line_straight(int move_distance){
  int move_speed=60; // mm/s
  int move_time=move_distance*100/move_speed; // *100ms
  for (int i=0;i<move_time;i++){
        Desiredl = move_speed*1.8;
        Desiredr = move_speed*1.8;
        directl=1;
```

```
        directr=1;
        directll=-1;
        directrr=-1;
        delay(100);
    }
    Desiredl = 0;
    Desiredr = 0;
}
//-------------------------------------------------

void turn_angle_left(int angle){
  int move_speed=50; // mm/s
  int move_time=angle/90.0*5.97*100/move_speed; // *100ms
  for (int i=0;i<move_time;i++){
        Desiredl = move_speed*1.8;
        Desiredr = move_speed*1.8;
        directl=0;
        directr=1;
        directll=1;
        directrr=-1;
        delay(100);
    }
    Desiredl = 0;
    Desiredr = 0;
}
//-------------------------------------------------

void turn_angle_right(int angle){
  int move_speed=50; // mm/s
  int move_time=angle/90.0*5.97*100/move_speed; // *100ms
  for (int i=0;i<move_time;i++){
        Desiredl = move_speed*1.8;
        Desiredr = move_speed*1.8;
        directl=1;
        directr=0;
```

```
        directll=-1;
        directrr=1;
        delay(100);
    }
    Desiredl = 0;
    Desiredr = 0;
}
//--------------------------------------------------

void stop_motors(){
    MsTimer2::stop();
    analogWrite(Motor_right_PWM,0); // right motor
    digitalWrite(Motor_right_direction,0); //right
    analogWrite(Motor_left_PWM,0); // left motor
    digitalWrite(Motor_left_direction,0); //left
    while(1){
      delay(100);
    }
}



//--------------------------------------------------

// the loop function runs over and over again forever
void loop() {
  delay(1000);
    if(Mode == '0'){
      delay(100);
    }
    else if(Mode=='1'){
      line_straight(3);
    }
    else if(Mode=='2'){
     delay(3000);
```

```
    turn_angle_right(97);
    line_straight(11);
    turn_angle_left(97);
    line_straight(2);
   }
   else if(Mode=='3'){
    line_straight(3);
   }
   else if(Mode=='4'){
    stop_motors();
   }
}
```

B> ARDUINO UNO CODE FOR ONE OF THE NODES:

```
#include <Wire.h>
#include<SoftwareSerial.h>
SoftwareSerial Xbee (0,1);
char Mode = '0';

const int pingPinl = 13; // Trigger Pin of Ultrasonic Sensor
const int echoPinl = 12; // Echo Pin of Ultrasonic Sensor
const int pingPinc = 11; // Trigger Pin of Ultrasonic Sensor
const int echoPinc = 10; // Echo Pin of Ultrasonic Sensor
const int pingPinr = 9; // Trigger Pin of Ultrasonic Sensor
const int echoPinr = 8; // Echo Pin of Ultrasonic Sensor
long duration, inches, cmc,cml,cmr;

void setup() {
  Wire.begin();
  Serial.begin(9600); // Starting Serial Terminal
  Xbee.begin(9600);
}
```

```
long microsecondsToCentimeters(long microseconds) {
  return microseconds / 29 / 2;
}

int check(int centre, int left, int right){
 if(centre <=20) return '9';
 else if (left <=6) return '9';
 else if (right <=6) return '9';
 else return '8';

 }

 int check_forward(){
  pinMode(pingPinc, OUTPUT);
  digitalWrite(pingPinc, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPinc, HIGH);
  delayMicroseconds(10);
  digitalWrite(pingPinc, LOW);
  pinMode(echoPinc, INPUT);
  duration = pulseIn(echoPinc, HIGH);
  cmc = microsecondsToCentimeters(duration);
  return cmc;
  }

  int check_left(){
  pinMode(pingPinl, OUTPUT);
  digitalWrite(pingPinl, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPinl, HIGH);
  delayMicroseconds(10);
  digitalWrite(pingPinl, LOW);
  pinMode(echoPinl, INPUT);
  duration = pulseIn(echoPinl, HIGH);
```

```
  cml = microsecondsToCentimeters(duration);
  return cml;
  }

  int check_right(){
  pinMode(pingPinr, OUTPUT);
  digitalWrite(pingPinr, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPinr, HIGH);
  delayMicroseconds(10);
  digitalWrite(pingPinr, LOW);
  pinMode(echoPinr, INPUT);
  duration = pulseIn(echoPinr, HIGH);
  cmr = microsecondsToCentimeters(duration);
  return cmr;
  }

void loop() {
 Serial.println("------------------");
 Serial.println(Mode);
// Mode 0 sensors off
  if(Mode == '0')
  {
   char val = check(99,99,99);
   delay(500);
   Serial.println(val);
   if(Xbee.available() > 0)
   {
    char chc = '8';
    chc = Xbee.read();
    if (chc == '1'){
    Serial.println("received change trigger");
    Mode = '1';
    }
```

```
   }
   }
// Mode 1 forward motion - front sensor
  else if(Mode == '1')
   {
    int centersns = check_forward();
    char val = check(centersns,99,99);
    delay(500);
    Serial.println(val);
    if(Xbee.available() > 0)
    {
     char chc = '8';
     chc = Xbee.read();
     if (chc == '2'){
     Serial.println("received change trigger");
     Mode = '2';
     }
    }
   }
// Mode 2 transformation - all off
   else if(Mode == '2')
   {
    char val = check(99,99,99);
    delay (500);
    Serial.println(val);
    if(Xbee.available() > 0)
    {
     char chc = '8';
     chc = Xbee.read();
     if (chc == '3'){
     Serial.println("received change trigger");
     Mode = '3';
     }
    }
```

```
   }
// Mode 3 left and right
   else if(Mode == '3')
   {
 //   int leftsns = check_left();
  //  int rightsns = check_right();
    char val = check(99,99,99);
    delay(500);
    Serial.println(val);
    if(Xbee.available() > 0)
    {
     char chc = '8';
     chc = Xbee.read();
     if (chc == '4'){
     Serial.println("received change trigger");
     Mode = '4';
     }
    }
   }
//Mode 4 transformation - long
    else if(Mode == '4')
   {
    char val = check(99,99,99);
    delay(500);
    Serial.println(val);
    if(Xbee.available() > 0)
    {
     char chc = '8';
     chc = Xbee.read();
     if (chc == '5'){
     Serial.println("received change trigger");
     Mode = '5';
     }
    }
```

```
  }

 //Mode 5 - forward
   else if(Mode == '5')
  {
   int centersns = check_forward();
   char val = check(centersns,99,99);
   delay(500);
   Serial.println(val);
   if(Xbee.available() > 0)
   {
    char chc = '8';
    chc = Xbee.read();
    if (chc == '6'){
    Serial.println("received change trigger");
    Mode = '6';
    }
   }
  }

  else if(Mode == '6')
  {

  }
  else
   Serial.println('x');

  Wire.beginTransmission(9); // transmit to device #9
  Wire.write(Mode);
  Wire.endTransmission();
  delay(200);

}
```

C> SERVER CODE FOR ARDUINO:

```
#include<SoftwareSerial.h>
SoftwareSerial Xbee (0,1);
char Mode = '0';
void setup() {
  Serial.begin(9600);
  Xbee.begin(9600);
}

void loop() {
switch(Mode){
  case '0':
  Serial.println('0');
  delay(1000);
  Serial.println('1');
  delay(2000);
     Mode = '1';
  break;

  case '1':
  if(Xbee.available()>0){
   char obs = '8';
   obs = Xbee.read();
   if(obs == '9'){
     Serial.println('2');
     delay(1000);
     Mode = '2';
     }
   }
  break;

  case '2':
     delay(3000);
     Serial.println('3');
```

```
   delay(1000);
   Mode = '3';
break;

case '3':
if(Xbee.available()>0){
 char obs = '8';
 obs = Xbee.read();
 if(obs == '9'){
  Serial.println('4');
  delay(1000);
  Mode = '4';
  }
 }
break;

case '4':
 delay(10000);
 Serial.println('5');
 delay(1000);
   Mode = '5';
break;

case '5':
 delay(10000);
 Serial.println('6');
 delay(1000);
  Mode = '6';

case '6':
 delay(10000);
 Serial.println('6');
 delay(1000);
 Mode = '6';
```

```
  break;
}
}
```

D> ARDUINO CODE FOR TESTING MONA'S PROXIMITY SENSORS:

```
// LED
int LED1 = 13;
int LED2 = 12;
int IR_enable=4;
int IR_threshold= 700;

void setup() {
// initialize serial communication at 9600 bits per second:
 Serial.begin(9600);
// initialize Ports
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(IR_enable, OUTPUT);
}

// Variables for 5 IR proximity sensors
int IR_right,IR_right_front,IR_front,IR_left_front,IR_left;

void IR_proximity_read(){    // read IR sensors
  int n=5;  // average parameter
  digitalWrite(IR_enable, HIGH);  //IR Enable
  IR_right=0;
  IR_right_front=0;
  IR_front=0;
  IR_left_front=0;
  IR_left=0;
  for (int i=0;i<n;i++){
    IR_right+=analogRead(A3);
```

```
    IR_right_front+=analogRead(A2);
    IR_front+=analogRead(A1);
    IR_left_front+=analogRead(A0);
    IR_left+=analogRead(A7);
    delay(5);
  }
  IR_right/=n;
  IR_right_front/=n;
  IR_front/=n;
  IR_left_front/=n;
  IR_left/=n;
}

// send IR readings to Serial Monitor
void Send_sensor_readings(){
 Serial.print("Right: ");
 Serial.print(IR_right);
 Serial.print('|');
 Serial.print("     Right_fr: ");
 Serial.print(IR_right_front);
 Serial.print('|');
 Serial.print("     Front: ");
 Serial.print(IR_front);
 Serial.print('|');
 Serial.print("     Left_fr: ");
 Serial.print(IR_left_front);
 Serial.print('|');
 Serial.print("     Leftt: ");
 Serial.println(IR_left);
}

void check(){
  if (IR_front<IR_threshold)
  digitalWrite(LED1,HIGH);
```

```
  else
  digitalWrite(LED1,LOW);
}
void loop() {
  // put your main code here, to run repeatedly:
 IR_proximity_read();
 Send_sensor_readings();
 check();
}
```

E> ARDUINO CODE FOR TESTING HC-SR04 ULTRASONIC SENSORS:

```
#include <Wire.h>
#include<SoftwareSerial.h>
SoftwareSerial Xbee (0,1);
char Mode = '0';

const int pingPinl = 13; // Trigger Pin of Ultrasonic Sensor
const int echoPinl = 12; // Echo Pin of Ultrasonic Sensor
const int pingPinc = 11; // Trigger Pin of Ultrasonic Sensor
const int echoPinc = 10; // Echo Pin of Ultrasonic Sensor
const int pingPinr = 9; // Trigger Pin of Ultrasonic Sensor
const int echoPinr = 8; // Echo Pin of Ultrasonic Sensor
long duration, inches, cmc,cml,cmr;



//setup
void setup() {
   Wire.begin();
   Serial.begin(9600); // Starting Serial Terminal
   Xbee.begin(9600);
}

long microsecondsToCentimeters(long microseconds) {
```

```
  return microseconds / 29 / 2;
}
int check(int centre, int left, int right){                    //status check function
 if(centre <=10) return 1;
 else if (left <=6) return 1;
 else if (right <=6) return 1;
 else return 0;
 }
//center sensor
 int check_forward(){
  pinMode(pingPinc, OUTPUT);
  digitalWrite(pingPinc, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPinc, HIGH);
  delayMicroseconds(10);
  digitalWrite(pingPinc, LOW);
  pinMode(echoPinc, INPUT);
  duration = pulseIn(echoPinc, HIGH);
  cmc = microsecondsToCentimeters(duration);
  return cmc;
  }
//left sensor
  int check_left(){
  pinMode(pingPinl, OUTPUT);
  digitalWrite(pingPinl, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPinl, HIGH);
  delayMicroseconds(10);
  digitalWrite(pingPinl, LOW);
  pinMode(echoPinl, INPUT);
  duration = pulseIn(echoPinl, HIGH);
  cml = microsecondsToCentimeters(duration);
  return cml;
  }
```

```
//right sensor
 int check_right(){
   pinMode(pingPinr, OUTPUT);
   digitalWrite(pingPinr, LOW);
   delayMicroseconds(2);
   digitalWrite(pingPinr, HIGH);
   delayMicroseconds(10);
   digitalWrite(pingPinr, LOW);
   pinMode(echoPinr, INPUT);
   duration = pulseIn(echoPinr, HIGH);
   cmr = microsecondsToCentimeters(duration);
   return cmr;
   }

void loop() {
//  Wire.beginTransmission(9); // transmit to device #9
//  Wire.write(val);
//  Wire.endTransmission();
  Serial.print("forward:");
  Serial.print(check_forward());
  Serial.print("----");
  Serial.print("left:");
  Serial.print(check_left());
  Serial.print("----");
  Serial.print("right:");
  Serial.println(check_right());
//  Serial.println(check_left());
//  Serial.println(check_right());
  delay(200);

}
```

F> MATLAB SIMULATION CODE:

```
t=0.2;
%circle vals
%cirlce
x1=30;
y1=0;
r=15;
th = 0:pi/100:2*pi;
a = r*cos(th)+x1;
b = r*sin(th)+y1;
x = [-1.5,-1.5,-1.5,-1.5,-0.5,-0.5,-0.5,-0.5,0.5,0.5,0.5,0.5,1.5,1.5,1.5,1.5];
y = [-1.5,-0.5,0.5,1.5,-1.5,-0.5,0.5,1.5,-1.5,-0.5,0.5,1.5,-1.5,-0.5,0.5,1.5];
plotxy(x,y)
pause(t)

%expand to 2x / x=1
for i=1:0.25:2
   x = i.*x;
   y = i.*y;
   plotxy(x,y)
   pause(t)
   x = (1/i).*x;
   y = (1/i).*y;
end
%reset value to max
x = 2.*x;
y = 2.*y;

% intial formation x-x=1/y-y=1/robots capable of moving 0.25-0.5 per step
% Start transformation
% C1 and C4 move up by y=1

for loop = 1:1:4
```

```
    for i = 1:1:16
      if (x(i) == -3 ||x(i) == 3)
        y(i)=y(i)+0.25;
      end
    end
 plotxy(x,y)
 pause(t)
end

% C1 and C4 move inwards

for loop = 1:1:8
    for i = 1:1:16
      if (x(i) < -1)
        x(i)=x(i)+0.5;
      end
      if (x(i) > 1)
        x(i)=x(i)-0.5;
      end
 end
 plotxy(x,y)
 pause(t)
end

% Expand vertically
for i=1:0.25:2
    y = i.*y;
    plotxy(x,y)
    pause(t)
    y = (1/i).*y;
end
%reset value to max
y = 2.*y
```

```
% C1 moves y=y-1 down
for loop = 1:1:4
   for i = 1:1:16
     if (x(i) == -1)
       y(i)=y(i)-0.25;
     end
   end
 plotxy(x,y)
 pause(t)
end

% C1 moves inwards towards C2
for loop = 1:1:8
   for i = 1:1:16
     if (x(i) < 1)
       x(i)=x(i)+0.5;
     end
 end
 plotxy(x,y)
 pause(t)
end

%move towards target
flag =1;
while(flag == 1)
 for i = 1:1:16
     if (x(i) < a -1 )
       x(i)=x(i)+0.5;
     else
       flag = 0;
     end
 end
 plotxy(x,y)
 pause(t)
```

```
end
%start wraping around
%reset flag
flag=1;


for iloop=1:1:200
   for jloop=1:1:200
      if(a(iloop)<=18 && b(jloop)>-10 && b(jloop)<10)
         while(flag==1)
           for i = 1:1:16
            if (x(i) < a(iloop) -0.2 && y(i)> b(jloop)-0.1 && y(i)<b(jloop)+0.1 )
              x(i)=x(i)+0.1;
            else
              %flag = 0;
            end
           plotxy(x,y);
           pause(0.1);
          end
         end

      end
   end
end
```

G> PLOTXY FUNCTION FOR MATLAB:

```
function [] = plotxy(x,y)
plot(x,y,'r.','MarkerSize',10)
xlim([-20,20])
ylim([-20,20])
hold on;

%cirlce
x1=30;
```

```
function [] = plotxy(x,y)
plot(x,y,'r.','MarkerSize',10)
xlim([-20,20])
ylim([-20,20])
hold on;

%cirlce
x1=30;
y1=0;
r=15;
th = 0:pi/100:2*pi;
a = r*cos(th)+x1;
b = r*sin(th)+y1;
plot(a,b);
xlim([-20,20])
ylim([-20,20])

%line
xline = [18,18];
yline = [-20,20];
plot(xline,yline);

hold off

end
```

H> PLOT FUNCTION FOR MATLAB:

```
function [] = plotxy(x,y)
plot(x,y,'r.','MarkerSize',10)
xlim([-20,20])
ylim([-20,20])
hold on;
```

```
function [] = plotxy(x,y)
plot(x,y,'r.','MarkerSize',10)
xlim([-20,20])
ylim([-20,20])
hold on;

%cirlce
x1=30;
y1=0;
r=15;
th = 0:pi/100:2*pi;
a = r*cos(th)+x1;
b = r*sin(th)+y1;
plot(a,b);
xlim([-20,20])
ylim([-20,20])

%line
xline = [18,18];
yline = [-20,20];
plot(xline,yline);

hold off

end
```