

COLLABORATIVE ROBOT BEHAVIOUR

Daniel Saxil-Nielsen

BSc Artificial Intelligence and Cybernetics, siu04dts@rdg.ac.uk

Abstract - This paper addresses the problem of mobile robots executing complex group behaviours in an attempt to form collaborative group behaviour. The paper will first look at forming a set of basic behaviours. These behaviours will then be evolved into more complicated group behaviours such as flocking. This is done in order to solve specific problems relating to goal finding and obstacle avoidance within a real world environment. This is completed without the need for inter-robot communication. This provides a behaviour which mimics natural flocks, as well as providing individual self-sufficient agents which aren't reliant on external influences.

Contents

1. Introduction	3
1.1 Deliverables	3
2. Research	3
2.1 Literary Review	3
2.2 Behaviour	4
2.3 Multi-agent Systems	4
2.5 Collaboration	4
2.6 Communication	4
2.7 Flocking	5
2.8 Domain Restrictions	5
3. Robot Architecture	5
3.1 Control Architecture	6
3.2 Software Architecture	6
3.2.1 <i>Decision Layer</i>	6
3.2.2 <i>Code Explanation</i>	7
3.3 Hardware Architecture	9
3.3.1 <i>Power</i>	9
3.3.2 <i>Chassis</i>	9
3.3.3 <i>Movement</i>	10
3.3.4 <i>Obstacle Avoidance</i>	10
3.3.4.1 <i>Ultrasonic Transmitter</i>	10
3.3.4.2 <i>Ultrasonic Receiver</i>	11
3.3.5 <i>Localisation</i>	11
3.3.5.1 <i>Phototransistors</i>	12
3.3.2.2 <i>LED's</i>	13
3.3.6 <i>Electronics and PCB Design</i>	13
3.3.7 <i>Hardware Limitations</i>	13
3.3.7.1 <i>Daylight</i>	13
3.3.7.2 <i>Ultrasonic Transducers</i>	13
3.3.7.3 <i>Movement</i>	14
4. Algorithms	14
4.1 Exploring	14
4.2 Homing	15
4.3 Following	15
4.4 Simulations	17
5. Results	18
5.1 Analysis of Behaviour	18
5.2 Flocking	18
5.2.1 <i>Leader Allocation</i>	18
5.2.2 <i>Algorithm</i>	19
5.2.3 <i>Performance</i>	19
6. Discussion	20
7. Further Work	20
7.1 Foraging	21
8. Conclusion	21
9. References	22
10. Appendix	23

1. Introduction

The goal of artificial intelligence (AI) is the effective and realistic simulation of natural intelligence. Through the creation of mechanisms to simulate these natural behaviours their complex nature can begin to be understood. One of the major underpinnings of natural behaviours is the ability for agents to perform a high level of inter-agent collaboration, with little or no communication. This paper intends to describe work which will investigate the effects of collaboration with multiple agents, within an environment which doesn't allow communication.

These ideas are based on the notion of collaboration, a means by which agents can interact to achieve a common goal, in our case the goal is for the agents to flock. This will be achieved through the introduction of basic behaviours [1], such as exploring, following and homing to achieve complicated and robust levels of interaction.

Previous attempts at flocking between mobile robots have all been carried out aided by communication [1], [2] and [14], whether it is direct or indirect communication. This paper postulates that effective levels of interaction can be achieved without the need for any direct communication and very little, if not any indirect communication, thus creating more reliable and faster agents which produce behaviour closer to that of natural intelligence.

1.1. Deliverables

The goal of this paper is to describe the methods used in order to achieve a level of collaboration between the agents, specifically to achieve a level of flocking with a number of key behaviours such as obstacle avoidance and goal finding. Each individual behaviour will be measured against a given set of criteria:

- Reliability – *how reliable is the behaviour?*
- Repeatability – *is this behaviour repeatable?*
- Scalability – *how is the behaviour affected by group size?*

Using the above criteria the performance of the behaviour can be evaluated and easily compared, each behaviour would be implemented into multiple homogenous mobile robots specifically designed for this paper. Each robot, agent, will be given the ability to detect obstacles and the location of other agents, using these basic sensors, the agents will achieve an advanced level of collaboration.

2. Research

2.1. Literary Review

There has been a number of papers and thesis devoted to the subject of collaboration of multi-agent systems, the most predominant is "Interaction and Intelligent Behaviour" by Maja Mataric [1]. This thesis talks about increasing the level of complexity of multi-agent systems. It proposes increasing both the cognitive and environmental complexity of such systems, in an attempt to further understand these systems. Specifically the author is involved in combining basic behaviours to produce more advanced behaviours, such as flocking and foraging. The paper concludes that the thesis is only the "*foundation in a continuing effort towards studying more complex behaviour, and through it, more complex intelligence*". The thesis draws

interesting ideas and points such as the combination of basic behaviours and well as results incorporating ideas such as Q-learning.

The original paper which first brought flocking to the forefront of multi-agent research was “Flocks, Herds, and Schools: A Distributed Behavioural Model” by Craig Reynolds [3]. This paper was one of the first to simulate flocking and describe, in great detail, the theory and principles associated with creating a successful flocking motion. Even though this paper was written with simulation in mind it still gives the idea of how a flock works as well as the more particle applications of flocking.

2.2. Behaviour

The term behaviour is used widely throughout A.I., and more specifically throughout this paper. While researching for this paper a number of subtly different definitions were coined for behaviour, but what is behaviour? For the purpose of this paper we define behaviour as a *means to reach a certain goal*. For example if a Sea Otter wants to open a mussel it will use a rock to hit the mussel. The goal is to open the mussel and the behaviour is hitting the mussel with the rock. In our case the goal for the agents is to avoid obstacles by using the behaviour known as flocking.

2.3. Multi-agent System

A great deal of research has been carried out in multi-agent systems, but why do we need/use multi-agent systems? The answer lies in particle swarm optimisation [11]. A group of agents can solve a problem faster than a single agent. In our case, as mentioned above, the goal is for the agents to move while avoiding obstacles this can be easily accomplished in the behaviour known as a flock because each member of the flock is protected by the other members.

Multi-agent systems also have the characteristic of having a high level of redundancy. For example if a single agent experiences a malfunction, then only that agent will be effected leaving the other agents to continue until the goal is reach. This has great advantages over single more complicated robots which have a higher chance of malfunction than smaller multiple robots. Even though the relative cognitive complexity of the robots are quite small the benefits of multi-agents is the combined congestive complexity is extremely high.

2.4. Collaboration

Collaboration is a key concept of this paper. Typically, collaboration is the combined working of groups to accomplish a given goal. Collaboration is an advanced form of interaction which is also a form of behaviour. The point at which interaction becomes collaboration is difficult to determine. Interaction does not only happen between agents but also any given agent and its surrounding environment. Therefore the point at which interaction becomes collaboration is the point where the agent chooses to act with the group instead of its own sensory information.

2.5. Communication

Communication is a difficult and complicated topic in multi-agent systems, there has been much debate about the level of communication that should be allowed between such systems. Papers such as [4], [15], state that the communication language is not fixed and provides a

method where the language can evolve over time, where others [5], [14], use either direct or indirect communication to produce the desired results.

There are two main forms of communication, either direct or indirect. Firstly direct communication is the method by which information is passed between robots on a robot by robot basis, meaning there is an intended recipient. This can be information such as speed, heading etc. In contrast, indirect communication is the method by which communication is passed through changes in the environment.

In this paper the amount of communication between agents will be strictly limited. There will be no form of direct communication either between agents or with an external source. There will also be very little indirect communication; the behaviours of the agents will be controlled solely through the direction in which the nominated leader will be heading.

2.6. *Flocking*

The most predominate species which flock are birds, however flocking is not just restricted to birds, all animals can exhibit the structured group movement known as a flock. Throughout evolution flocking has been a popular way to protect groups from predators, increasing the search capacity for food as well as the social and mating advantages that groups bring.

Flocking is a form of emergent behaviour, first simulated in 1986 by Craig Reynolds [3], it has been widely recognised and improved upon [16]. It forms a simple, fast and effective way of searching a multi-dimensional domain. Flocking is made up of three basic behaviours [3]:

- Aggregation – *Steer towards agents.*
- Separation – *Keep a minimum distance from other agents.*
- Cohesion – *Move towards the average position of local agents*

With the successfully combination of these three behaviours a level of flocking can be achieved between multi agents within the simulation. The *boids* paper also postulates more complicated set of rules including strong wind actions, speed limitation, landing and the ability to switch off the flock. However because the *boids* paper focuses on the simulation it takes great liberties compared with natural flocks. For example the paper works out the where each agent should fly by calculating the centre of mass of the neighbouring agents. This is of course impossible in nature.

Emergent behaviours in robotics, however, are far more difficult problems to solve because like animals the agents do not have a global view, unlike a simulation. Flocking is an especially complicated behaviour because it requires both aggregation and dispersion, in other words maintain a fixed distance from an obstacle, without colliding with that obstacle.

For a successful flock a leader is required. This leader has to be dynamic and not pre-programmed. In our case our leader will first be selected on a *first on* principle, whichever agent is first on will be the leader. The leader will then alternate depending on the environmental circumstances of the current leader. There can however be more than one leader in control of more than one flock at any one time. When these two leader encounter each other one will lose there leadership and merge into the other flock.

2.7. Domain Restrictions

In order to restrict the number of variables and increase the constraints a number of domain restrictions were imposed for this paper. Firstly all the agents are homogenous. This increases the robustness of the agents and does not require any particular agent to accomplish any given task. The homogeneity also makes the behaviour of the agent predictable leading to the ability to easily for each agent to react to another. The second domain restriction is the limitation of communication; no agent is allowed to communicate with another either directly or indirectly. Each agent will basically have the ability to ‘see’ other agents and react to the movements seen by an agent; this is explained in greater detail below. The third domain restriction is the need for an environment relatively low on noise, specifically directed light sources. Due to the functionality of the localisation system directed light sources cause a great deal of noise within agent. The domain is also restricted by the number of agents that can be involved in any given behaviour. To show the agents operating in a domain whereby they are no limited by these factors a computer simulation will be created to model the exact behaviour of the agents.

3. Robot Architecture

The robots which were constructed for the paper were originally inspired by the Mobile Robot module at the University of Reading [6]. These were originally designed by Lawrence Withers, however this design was very limited for the applications in hand and thus had to be expanded upon. Specifically the current microcontroller (PIC16F87) had to be upgraded in order to accomplish the goals set out for this paper. A new chassis design was needed to cope with the extra hardware, specifically relating to the mounting of sensors. New printer circuits boards and circuit diagrams had to be developed to successfully integrate the new features that are required for this paper into the model.

Each of the robots created for this paper has a modular approach to the design. These robots consist of 3 major and interchangeable modules, motor drivers, processor, and localisation. This allows for a robust framework which can easily be modified and improved upon at a later date.

3.1. Control Architecture

Each agent is controlled through a PIC microcontroller, PIC16F690. The PIC16F690 is a low pin count microcontroller, which features 18 general I/O pins, 2 comparators as well as an 8 MHz internal oscillator. This microcontroller is a robust, low cost approach to the control architecture of the agents. The microcontroller provides the means to control and programme the agent, this makes the agents self sufficient and not reliant on external computing power. The PIC16F690 can be programmed in both C and Assembler through the MPLAB IDE environment; this program can then be downloaded through the Microchip PIC Kit2. The current set up requires the microcontroller to be removed and inserted into the PIC Kit demo board. However both PIC Kit2 and the PIC16F690 support the ability for onboard programming.

Figure 1 shows the pin out diagram for the 16F690 microcontroller. Out of the 18 inputs and outputs of the microcontroller 7 are used for obstacle avoidance, 4 are used for localisation, 4 are used to drive the motors and 2 for general purpose signalling lights. The microcontroller is mounted on a custom printer circuit board interfaced with the other two

modules of the agent. The control architecture has been designed so that the modules are ‘hot-swappable’.

20-pin PDIP, SOIC, SSOP

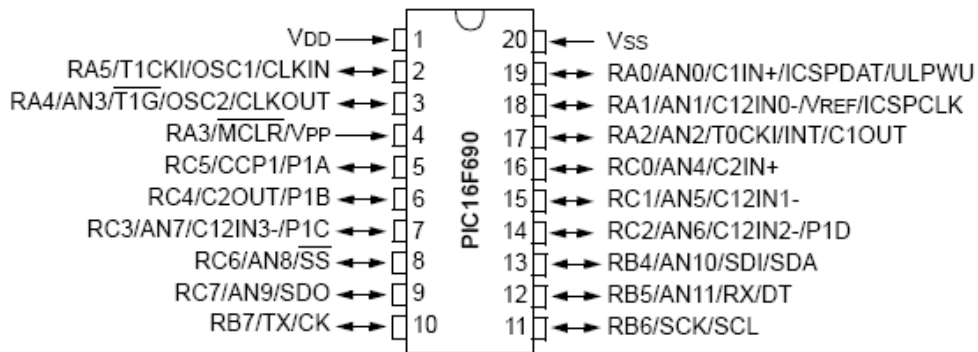


Figure 1, Pin out diagram of the 16F690 microcontroller [7]

Table 1 shows the current module set up for the agents designed for this paper. The modular approach has a number of significant advantages, the greatest being the ability to continuously develop individual modules. The following table outlines the current version of the models that were developed for this paper:

Module	Version
Motors	2
Microcontroller	4
Ultrasonic	2
Localisation	5

Table 1. Current version numbers of mobile robots.

3.2. Software Architecture

Each agent has a homogenous approach to its software architecture. The agent employs a subsumption like architecture consisting of a single fully reactive layer [8], with the ability for a decision layer; each agent makes decisions based on real time sensory information. This real time performance is achieved through a purely reactive layer which makes decisions on a set of pre-programmed commands. This was decided upon because it leads to a very small processing overhead and allows the agents to move fast and efficiently. Whereas planner system require large amounts of computing power to successful plan their route, they also cannot cope with fast changes in the environment such as other agents.

The software is mostly written in C and compiled by HiTech PICLite C compiler. The software is split into a number of procedures and functions which accomplish certain goals of the robot, for example to control the pulse width modulation of the motors. However the procedures and functions also convert the sensory information, such as ultrasonic timings into a form which can be easily read and incorporated into other parts of the programming. Each agent individually has a low cognitive ability but as with all multi-agent systems the combination of a large number of low ability agents will be able to solve a problem of great complexity.

3.2.1. Decision Layer

Originally the agents were designed with a purely reactive bottom up approach to there software architecture. However due to the hardware limitations found through experimenting, sections 3.3.7, a gap in the ultrasonic sensors was found. This dead spot can be solved programmatically through the introduction of a decision layer. This decision layer will search the surrounding space every x number of seconds to give the agent a picture of it's surrounding. The agent can then make a decision based on this information so that it can avoid areas where that dead spot would be exacerbated.

This is accomplished through the following algorithm:

```

If (counter == 3)
{
    for (int I = 1; I < 4: I++)
    {
        Turn(90);
        List[I].Range = Ping_sonar(); List[I].Angle = 90 * I;
    }
    List.Range.Sort();
    Turn(List[1].Angle);
}

```

Algorithm 1. Decision layer.

3.2.2. Code explanation

Lawrence Withers code for the Mobile Robot project [6] was available and was used as a reference point for the development of the code as many of the same tasks, specifically obstacle avoidance, will carried out. However the code did have to be heavily modified to cope with the change in the PIC microcontroller. There have been multiple versions of the code experimenting with all different aspect of what the microcontroller is capable of accomplishing. A great deal of knowledge was acquired through the PIC Kit2 tutorials which helped with the basic functionality of the microcontroller and the development environment.

The following functions describe the basic operation of the PIC microcontroller which controls the movement and sensory information of the agent. Additional behavioural functions are described later on. The first function is the interrupt within this the PWM duty cycles are set as well as initialising the global timer variables. After the motors are set up there are 5 functions to control the ultrasonic sensors. The first *sonar_task* sets up the ultrasonic transducers so they are enabled at the correct times, this works by rotating which transmitter should be activated every 244 milliseconds. Every 244 milliseconds the function *ping_sonar* is then called which transmits the signal and listens for the interrupt flag of the comparator to go high, it is then able to work out how long the signal has taken. This function can then convert this time into a range varying from 0 to 99. To successfully transmit the signal the function has to call one it's child functions, *ping_sonar_aux*, which pulls the V_{ref} pin of the comparator to ground and the calls either *ping_sonar1* or *ping_sonar2* depending on whether the left or right sonar is to be transmitted. These two functions provide the differential voltage to the transmitters.

The localisation system is less complicated requiring only one setup function, *isAnother*, which individually checks all the incoming pins of the microcontroller to see if any return high indicating that there is another agent within range. The final function is the *main* function. This is where all the previously non-interrupt functions are declared and executed. This is also where the LED's are set up to enable both the localisation system and the visual display.

3.3. Hardware Architecture

3.3.1. Power

Each agent works through a 9 volt power supply. This is in the form of a NiMH rechargeable PP3 battery mounted onto the chassis. These batteries are recharged externally through a plug in mains charger. The agent also features a 5 volt regulator for the microcontroller and a number of other low powered semiconductors. There is also 9 volts running to the motors and localisation system to provide maximum power and range. Each agent has a battery life of approximately 30 minutes.

3.3.2. Chassis

The main robot features a strong and durable chassis, this chassis is made out of 3 sheets of circular steel. Each sheet is mounted 2.5cm above the last with threaded rod giving the robot its modular design. The chassis was designed with the help of a computer aided design (CAD) program, AutoCad LT. This enabled the chassis to be designed and developed to a high degree of accuracy. Three separate templates were produced, one for each module. These designs were then used to allow easy marking and construction. Each plate has a relatively low degree of complication with 3 major holes for the threaded rod and a number of smaller holes to mount the PCB. The motor module requires a slightly higher level of complication because it requires the motors to be precisely mounted at a 90 degree angle. For this a standard sheet of steel was angled at either end to provide the required 90 degrees, this was then attached to the bottom plate, as can be seen in figure 2.

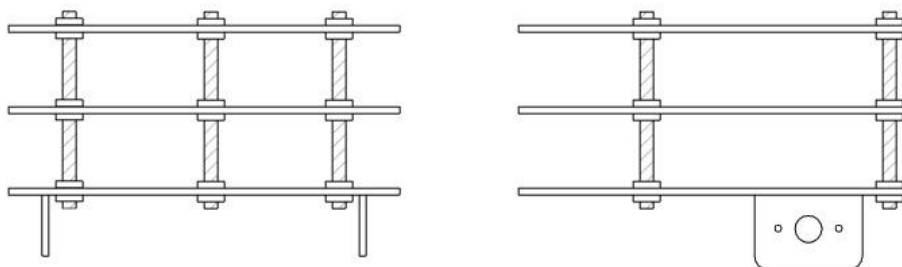


Figure 2, schematic design of chassis, front facing and left facing.

A number of different prototypes were built for the chassis. Figure 3, below shows the comparisons between the first prototype and the final design. The overall structure remains the same however the threaded rod, which holds the robot together, has been reorganised and thinner, whereas the cylindrical cap has been removed.

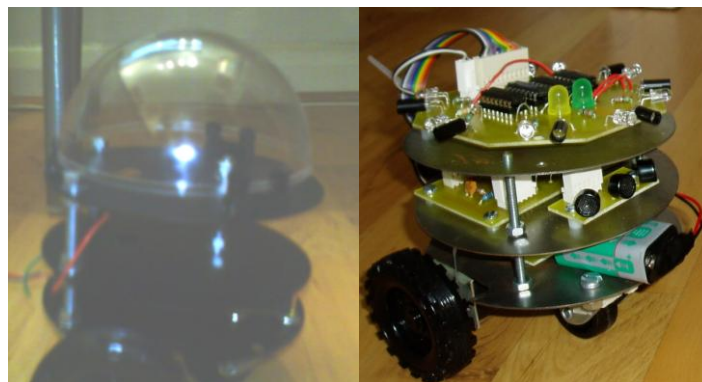


Figure 3, comparison of original chassis design to the final chassis design

3.3.3. Movement

Movement is accomplished through 2 differential driven 12volt D.C. motors. These are controlled through a high frequency pulse width modulation, PWM. PWM is effectively a square wave with a modulated duty cycle. A change in the duty cycle results in the average value of the waveform being changed; this can be given by equation 1.

$$a = \frac{1}{T} \int_0^T f(t) dt \quad (1)$$

Changing the average value of the waveform results in an increase or decrease, respectively, of voltage going through the motors, this leads to a variable speed. The PWM is used in conjunction with a semiconductor switch which controls the voltage or current across the motors. At any one time the amount of power being dissipated from the switch is the product of the current and voltage, thus because the switched are either high and have no voltage drop or low and have no current drain, the power dissipated by the switch is effectively nothing. In our case we are using a MOSFET (T4427) semiconductor switch, in combination with the software to produce a range of -8 through to 8 for the PWM.

Stability of the robot is achieved through the use of a single caster wheel mounted onto the chassis which provides free and easy movement of the robot. While there are 2 plastic wheels mounted onto the motors to provide both grip and momentum.

3.3.4. Obstacle Avoidance

The obstacle avoidance works by utilising 3 ultrasonic transducers, two transmitters and one receiver. This very efficient and advanced form of obstacle avoidance allows an object to be detected and avoided before the agent encounters the object.

3.3.4.1. Ultrasonic Transmitter

Each transmitter is fired with +5v and -5v providing the transducer with a 10v differential which dramatically increases the range. This is achieved through the microcontroller, by connecting both the positive and negative rails of the ultrasonic transducer, as shown in figure 4. The microcontroller can pull RC0 high and RC1 low, then reverse the polarities, this leads to the waveform shown in figure 4.

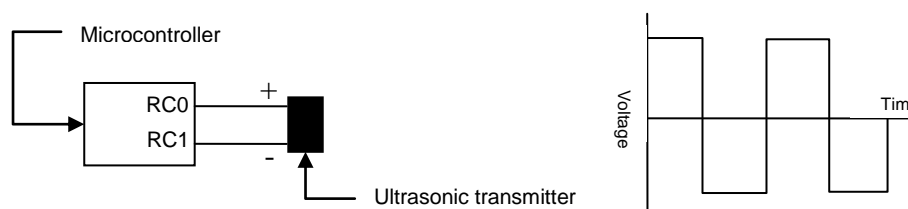


Figure 4. Diagram of ultrasonic transmitter and wave form applied to ultrasonic transmitter.

We use the assembler language to accurately control the timing of the transmissions. The frequency of the transmissions through the ultrasonic transducers is 40 KHz. Each transmitter is fired alternatively; this avoids the possibility of the two signals interfering. The microcontroller listens and waits until the signal has been received before sending out the new signal. If the signal is not received within a certain time the signal is declared lost.

3.3.4.2. Ultrasonic Receiver

The short sound wave that is emitted by the transmitters is deflected by obstacles back into the receiver. The received signal is then amplified and passed into the inbuilt comparator of the microcontroller. When the V_{ref} voltage to exceeds the V_{in} voltage the comparator fires, setting the interrupt flag of the comparator (CM1CON0) high. The time between the signal being transmitted and the interrupt flag going high, t , is measured. Utilising the Time of Flight [9] method the exact distance from the obstacle can be calculated.

$$d[m] = v \left[\frac{m}{s} \right] \times t[s] \quad (2)$$

If the V_{in} of the comparator is pulled low then high again, this causes the capacitor to discharge and recharge. This produces a range-swept gain [6]. If the ultrasonic ping is received faster it should have a higher voltage than that of a signal received later. Therefore to compensate for the power fall off we want the V_{ref} voltage to increase over time making the receiver more sensitive. We can see this in figure 5; V_{ref} has a slope proportional to $\frac{1}{r^2}$.

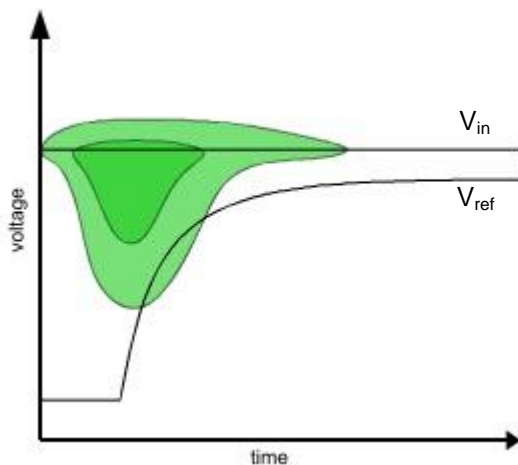


Figure 5. Graph showing the range-swept gain of the receiving circuit. The V_{in} shows the voltage of the ultrasonic receiver.

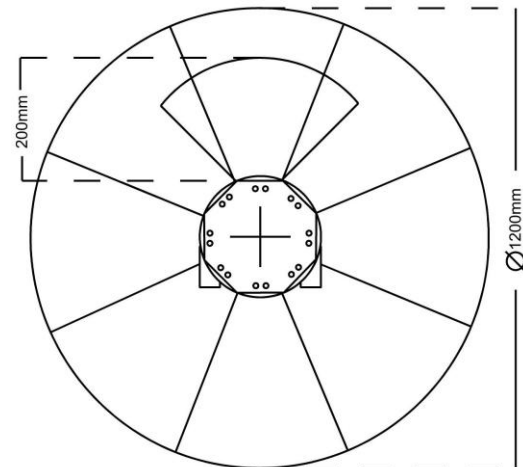


Figure 6. Schematic diagram of agent sensing range. Localisation, larger circle while ultrasonic range is the smaller arc.

When an object is closer a higher received voltage is generated forcing V_{ref} to exceed V_{in} which fires the comparator. The software architecture of the agent then measures the time between the signal being sent and received and converts this into a range between 0 and 99.

3.3.5. Localisation

Localisation is achieved through 8 phototransistors and high intensity light emitting diodes mounted 45 degrees apart on the top of each agent, as shown in figure 6. These diodes are an active target [9], as opposed to a passive target. These give the agent a detection range of approximately 600mm. Each agent has a bias towards the front this makes the front more sensitive to changes in light source than the rear. Utilising this method the agent is able to detect a light source in any one of the 8 phototransistors and rotate the agent until the centre front detector is facing the light source. The ultrasonic transducers are then activated in order

to compute the range of the light source. This method allows simple but effective inter-agent localisation.

The use of this method in localisation is very efficient. Instead of relying on inter-robot direct communication this method is fast and allows the agents to move at high speeds without lag. No other form of information is passed between agents; each agent knows the direction and speed of another agent through tracking its movements. This agent can then make a decision based on the information that is gathered.

3.3.5.1. Phototransistors

Each of the eight phototransistors used in the localisation system are extremely sensitive linear NPN transistors. With a 420nm to 1130nm photosensitivity range they are perfect for this application. Each phototransistor is connected to a threshold comparator, figure 7. This threshold comparator compares the incoming signal from the phototransistor with a reference signal, firing when the reference voltage exceeds the input voltage.

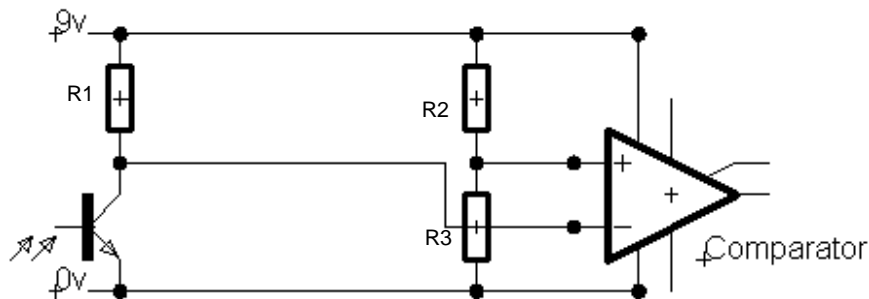


Figure 7. Threshold comparator.

By changing the resistance of R1 we are able to alter the gain of the phototransistors and thus alter the sensitivity of the phototransistor. By increasing R1 we will get a greater sensitivity but a much lower response time. Through a great deal of experimentation the optimum R1 value was 500kΩ whereas the resistors for the potential divider (R2 and R3) were 47kΩ and 33kΩ respectively. Each of the 8 phototransistors is connected to one of two LM348N which is a quad op-amp semiconductor, providing all 8 phototransistors with their own threshold comparator. Each of the 8 outputs from the comparator is then fed into 4 OR gates, which compresses the 8 signals into 4, saving vital microcontroller inputs.

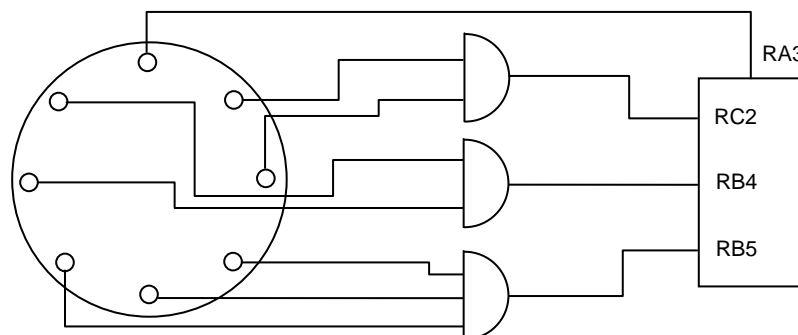


Figure 8. Schematic of which phototransistors and mapped to which inputs.

Figure 8 shows the mapping of phototransistors to their corresponding inputs. As you can see the back three phototransistors are combined into 1 this means that whichever one goes high the agent knows there is another agent behind and can rotate until the front facing signal phototransistor is facing the agent, this is also the case for the two phototransistors on either

side. However the front facing phototransistor is not connected to any OR gates to provide a straight heading.

3.3.5.2. LED's

There are 8 high brilliance LED's mounted next to each of the phototransistors, with a shield to avoid contamination between the light source and receiver. All 8 of the LED's are powered from the microcontroller through a transistor to provide them with maximum brightness.

3.3.6. Electronics and PCB Design

Electronic circuit design was carried out using Eagle Schematic designer, with provides easily editing and creation of electronic design. Each of the designs was tested on a breadboard, then on strip board before finalising the design. The design was then converted into a PCB design using the Eagle PCB editor. Each one of the modules has its own custom PCB design. Even though the Eagle PCB editor allows the schematics to be imported and auto-routed this produces dubious results and sometimes place signal lines next to power lines which can cause noise. Therefore all the tracks were joined by hand. This meant that first a number of prototypes had to be developed.

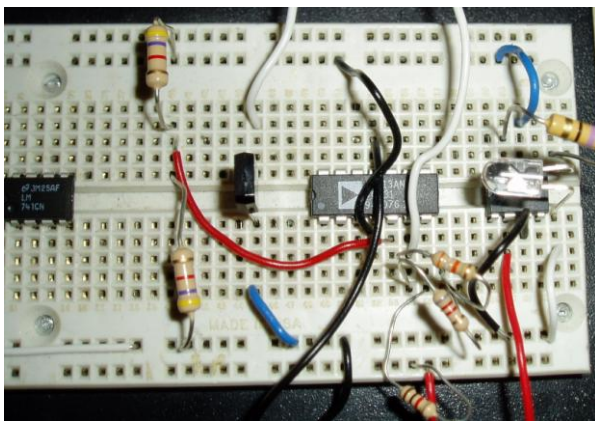


Figure 9. Prototype stage of the localisation board.

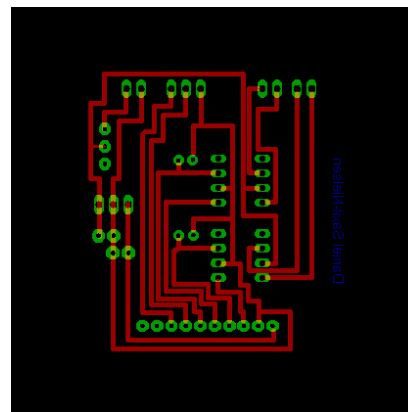


Figure 10. PCB layout of motor board.

3.3.7. Hardware Limitations

Hardware limitations impose restrictions on the agent's behaviours. In particular the types of behaviours the agents can achieve as well as the methods in which to control them. There are a number of major flaws with the hardware of each agent.

3.3.7.1. Daylight

The localisation system suffers from noise caused by daylight, or any other directed light source. This is caused by the fact that to achieve the range required the phototransistors have to be extremely sensitive. However this means that they are susceptible to all light sources and not just that of another agent. Attempts were made to have high and low pass filters to filter out daylight without any success.

3.3.7.2. Ultrasonic Transducers

The ultrasonic transducers suffer from two hardware limitations. The first being the blind spot in the ultrasonic transducers in the left hand side. This causes reflections in the sound wave when the agent approaches a wall at a specific angle making the sensors confused and possibly crash into that wall. The second is how the temperature affects ultrasonic sound waves. The speed of sound changes about 0.1 percent per degree of Fahrenheit. This corresponds to one millimetre error per degree of Fahrenheit, at one meter [10].

3.3.7.3. *Movement*

All attempts have been made to make sure each agent can move in a straight line. However due to restrictions with the hardware this is not always the case and can result in some agents going off course. However this should be compensated for in the behaviour of the agents.

4. Algorithms

As described in ‘Interaction and Intelligent Behaviour’ [1] basic behaviours can be combined to form more complex group behaviours. In this case flocking is made up of three basic behaviours, exploring, homing and following. This creates the problem of arbitration between different behaviours in order to produce the desired behaviour. Therefore the agents display mutually exclusive behaviours whereby two behaviours cannot happen at the same time. This will be dependent upon sensory inputs.

This section of the paper outlines the algorithms used to create these basic behaviours. Each algorithm was implemented in C and programmed into each agent to be tested. We will then compare each of the algorithms to the three criteria set out in the introduction:

- Reliability – *how reliable is the behaviour?*
- Repeatability – *is this behaviour repeatable?*
- Scalability – *how is the behaviour affected by group size?*

Using these three criteria we can judge the performance of each of the behaviour.

4.1. *Exploring*

Exploring is one of the simplest behaviours the agents can accomplish. It forms a vital and basic role of the agent. Exploring is done through the use of simple reactions to the ultra-sonic input, by determining the range of a given object the agent decides what to do. If an object is on the left it turns right and vice versa, the agent also has the ability to judge the severity of the turn by how close the object is. These both enable efficient exploration but also accurate obstacle avoidance.

The goal of exploring is to move while keeping a safe distance, d , away from obstacles. This can be expressed formally using equation 3, where R is the robot.

$$\forall R (Robot(R) \rightarrow explore (d > obstacle)) \quad (3)$$

Using equation 3 algorithm 2 can be derived.

```

if (object <= min_distance)
{
  if (object_left && object_right)
  {
    stop_motors;
    if (after 5 seconds)
      reverse and turn randomly;
  }
  else if (object_left) turn right;
  else if (object_right) turn left;
}
else wander;

```

Algorithm 2. General exploration and obstacle avoidance algorithm.

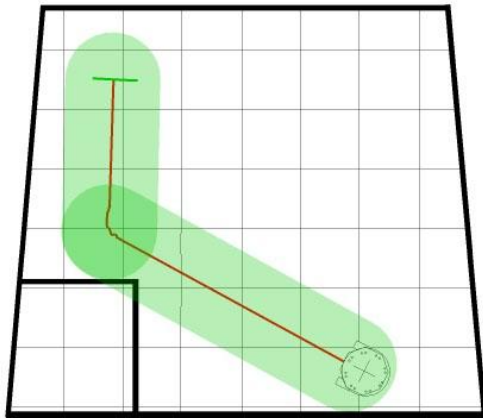


Figure 11. The agent exploring and avoiding an obstacle. Range of ultrasonic transducers is shown with the thicker path.



Figure 12. Photo of figure 11, showing the path taken.

Figure 11 shows the exploration and obstacle avoidance of an agent. The figure shows the detailed path of movement around the obstacle. The thicker path denotes the ultrasonic range. When the path collides with the obstacle the agent turns, avoiding that obstacle.

Through testing this algorithm it proved very reliable and repeatable. The algorithm is let down by the hardware limitations which were set out in section 3.3.7. When the agent approaches an obstacle on the left hand side at a less than 45 degree incline the obstacle is not seen and thus isn't avoided. To counteract this, smaller turning circles were introduced alongside increased sensor range which has almost eliminated the problem.

4.2. Homing

Homing is where one agent finds another agent in an attempt to decrease the distance between the agent and the goal location. Expressed formally, where P is position:

$$\forall R(\text{Robot}(R) \rightarrow \text{home}) \tag{4}$$

Where home is:

$$\left(\frac{dp_R}{dt} (p_R - p_{goal}) > 0 \right) \tag{5}$$

From these equations, 4 and 5; we can derive the following algorithm:

```

if not (object <= min_distance) | (goal is reached)
{
  if (signal on left) turn left;
  else if (signal on right) turn right;
  else explore();
}
    
```

Algorithm 3. Algorithm for homing.

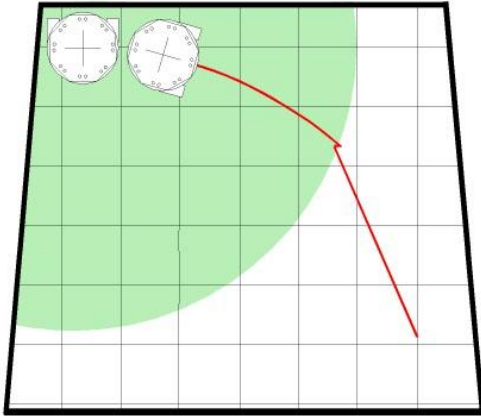


Figure 13. Agents homing in on a goal. The range of the goal is shown by the circle.



Figure 14. Photo of figure 13.

Figure 13 shows the implementation of the homing algorithm. In this example the second agent is the goal and the first agent’s objective is to find and home in on the goal. The area outlined by the circle is the range of the localisation system. Once the agent enters the range of this system it sees the other agent and homes in on it, otherwise the agent will just explore.

This algorithm also produced results which were both repeatable and reliable, proving that not only was algorithm correct but also the localisation system work effectively and quickly.

4.3. Following

Following is the most complicated out of the basic set of behaviours. The goal of following is to maintain a fixed distance between the follower, f , and the leader, l . This can be expressed formally by the following equation:

$$\exists l \left(\text{Leader}(l) \rightarrow \forall f \left(\text{Follower}(f) \rightarrow \frac{dp_f}{dt} (P_l - P_f) < m \right) \right) \quad (5)$$

The minimum distance between the leader and the follower is constrained by the ultrasonic sensors, whereas the direction of the leader is determined by the phototransistors.

The following algorithm can then be derived:

```
Home();
if (ultrasonic range < minimum) stop;
```

Algorithm 4. Following algorithm.

In this algorithm the goal is defined as being a set distance away from the leader. Since the leader is always moving the goal is never true, and thus the agent follows the leader while maintaining a fixed distance. If no leader can be found then the agent will explore, thus producing behaviour for both the leader and the follower.

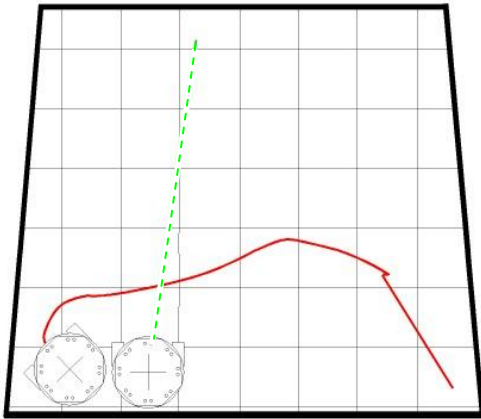


Figure 14. Following, leader denoted by dotted path



Figure 16. Photograph of figure 14.

Figure 14 shows the implementation of algorithm 3 in the agents. The dotted line denotes the leader whereas the solid line denotes the follower. This algorithm again proved reliable and repeatable. However due to the limitations of the number of agents created for this paper true scalability couldn't be accurately assessed. Therefore a simulation was created.

4.4. Simulations

A simulation is a very powerful tool in multi-agent robotics, because of the cost and time restraints that multi-agent robotics imposed it is very difficult to produce experiments with 1000's of robots, therefore we use simulations. The reason why this particular simulation was not carried out at the beginning was to evolve and create the correct algorithms for the real world and then impose these algorithms and restrictions in the simulated world where otherwise we could have taken liberties.

To build a simulation of group behaviour we need to incorporate the behaviours we have derived in the real world. Therefore we add algorithms to correspond to the sensors in the real world agents. This is done by using bounding spheres. A bounding sphere is an object a set radius from a centre point. We can test at any point if this bounding sphere is intersected by another sphere, thus we are able to determine whether the agent is close to other agents. Unlike the *boids* paper [3] we don't use procedures explicitly designed for software we need to convert the algorithms that were derived into a suitable form for the simulation. Therefore we use these bounding spheres to act as the localisation system and the ultrasonic system.

Using the new Microsoft XNA framework this simulation with 5 agents could be quickly and easily developed. The Microsoft XNA framework allows for simulations in both 3D and 2D and features easily manipulation of sprites. Five sprites were created in the simulation and each given two bounding spheres. The movement of these sprites is controlled

through the manipulation of the pixels. By continually incrementally increasing the pixel position of the agent and the bounding spheres we can move these agents. To produce turning circles we use a rotation variable that can be manipulated, and applied to these pixels using a sine or cosine function to give the appearance of the object turning.

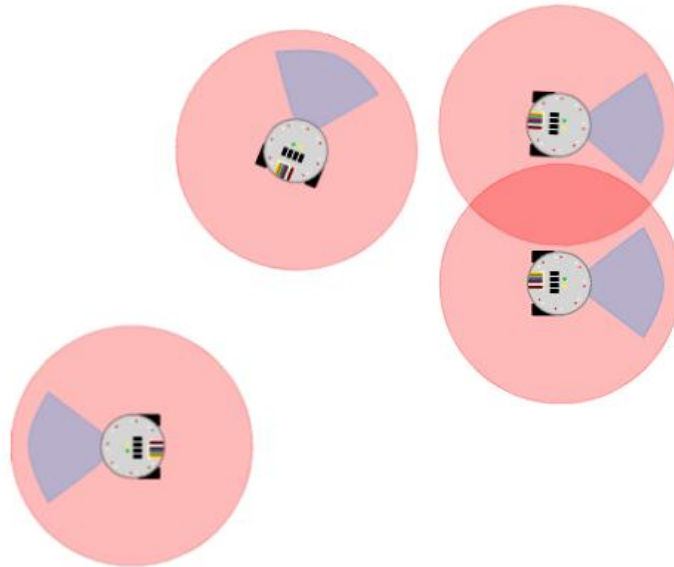


Figure 17. Mobile robot simulations using the XNA framework, red circles signify localisation range blue arcs signify ultrasonic range.

The same algorithms that were employed in the mobile robots were also employed in the simulation. This shows that each of the basic behaviours can be easily scaleable to larger groups.

5. Results

5.1. Analysis of Behaviour

The successful evaluation of behaviours relies on a key set of performance criteria. Early we set these criteria as 3 questions and compared each of the basic behaviours against these criteria. Each of the behaviours was also compared to a mathematical representation of the behaviour. However for more complicated sets of behaviours such as flocking and foraging more precise performance criteria needs to be set.

5.2. Flocking

As mentioned previously flocking is a form of structured group movement and aims to protect the members in the group and guide the group towards the goal. Flocking is the successful cohesion of the 3 basic behaviours. In this paper flocking is carried out by 2 homogenous mobile robots, each of the mobile robots has the 3 basic behaviours and reacts on sensory input to produce the desired behaviour. There is no direct communication between any of these agents, nor does the agent have any model of the world, it is merely acting on the distance and heading of other agents.

5.2.1. Leader Allocation

Since all the agents are homogenous in both their hardware and software, there is no predefined leader. The selection of the leader is determined by which agent is activated first. Agents become leaders if they are out of range of other agents. However leadership isn't fixed and can be swapped between agents. For example if agent1 is unable to move, leadership of the flock is passed to another agent, thus freeing agent1. This forms protective group behaviour, while still accomplishing the giving goal.

5.2.2. Algorithm

```

If another robot Follow() else
{
  explore();
  if (unable to explore) Follow();
}

```

Algorithm 5. Flocking.

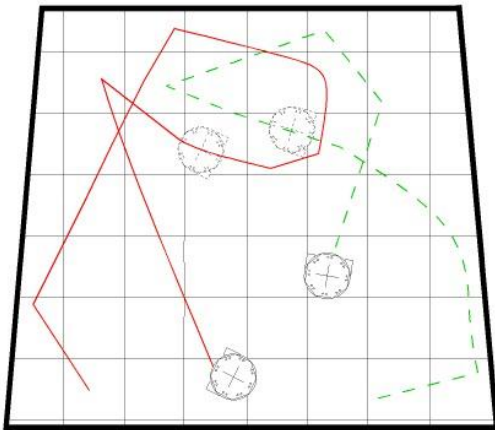


Figure 18. Implementation of the flocking algorithm.

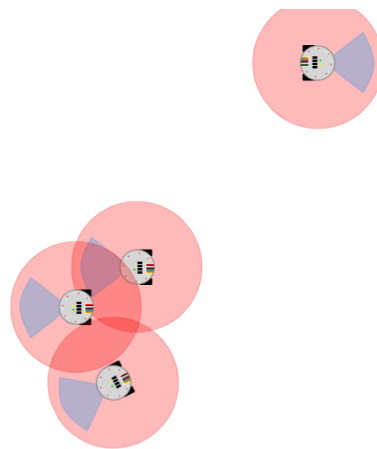


Figure 19. Flocking algorithm implemented in the simulation.

Figure 18 shows the implementation of the flocking algorithm. The paths shown are very erratic. However at key points the agents do show a flocking behaviour. This behaviour isn't sustainable though; there are two main reasons behind this. First the environment imposes a restriction which forces the agents to turn erratically; secondly flocks can prove unstable in small numbers. The larger the number of agents in a flock the more stable a flock becomes. This is proved with the simulation a larger number of agents within the simulation show a much higher chance of the flock being held. The simulation also shows that this algorithm is scalable up to 5 agents and both repeatable and reliable showing a constant behaviour.

5.2.3. Performance

The performance criterion for the flocking algorithm is the amount of time a flock formation can be held for. This is measured by taking 10 trials and timing the length that the flock is held, therefore timing how long until the leader swaps or until the obstacle avoidance becomes unsuccessful.

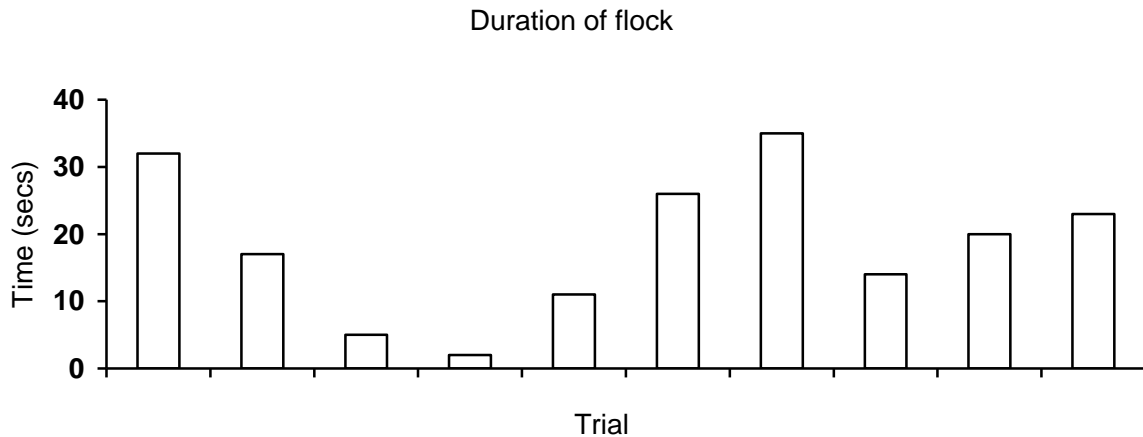


Figure 20. Graph showing the duration of the flock in seconds over 10 trials. Mean 18.5 Seconds

This can be seen in figure 20, it shows that the longest the flock could be successfully held was 36 seconds while the mean time is 18.5 seconds. The short length of time is mainly caused by the lack of agents in the flock and the enclosed area that the agents were tested. If we compare this with the results from the simulation:

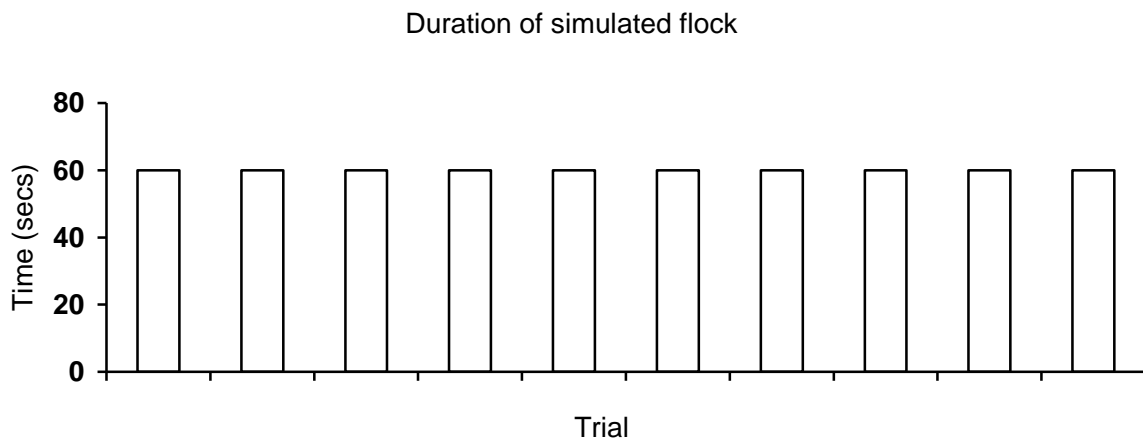


Figure 21. Graph showing the duration of the flock in seconds over 10 trials. Mean > 60 Seconds

All the trials produced results which were greater than 60 seconds. This proves that in a perfect environment the duration of the flock is not limited.

5.2.4. Evaluation

The flocking algorithm has proved very successfully and has proved both scalable and reliable. This is all achieved without any form of direct communication and fully reactive software architecture.

6. Discussion

This paper draws many areas of discussion most notably is the topic of communication. Communication within multi-agent systems is already an area of hot debate. This paper sets out to limit the amount of communication between such agents. The aim will be to have no communication between agents. However this leads to the question *is one agent following*

another is a form of indirect communication? On one hand the leader is changing the environment by moving, this causes the follower to react to the movement changes. However the leader isn't trying to communicate with the other robot, the robot is merely following a set of pre-programmed responses to its environment. Therefore is this direct communication? We would argue that it isn't, because unlike animals which explicitly use an indirect form of communication, such as ants to leave a trail for the other animals the agents are not trying to communicate with one another their actions are just reactions to changes in the environment.

The second area of discussion that this paper draws is the limitation of only creating 2 robots to test the theory in this paper. Ideally more robots would be created, however due to the complicated design and tested process each robot has to go through and the tight deadlines for this paper, it was not possible to create more. This is where the simulation is used, however the simulation doesn't provide a 100% accurate model of the real world robots, specifically the simulation has an ideal domain, whereas the robots have a limited domain. Therefore does the work in the paper accurately prove the goals set out in this paper? The paper proves that the goals work for two robots and using the information gathered from these two robots the simulation proves that the theory works for 5 robots, so yes the paper does accurately prove the goals.

7. Further Work

This paper is only a starting point of the field of multi-agent system, a great deal of work and research can be carried out to improve the results and carry out advanced forms of testing. This includes work such as increasing the number of agents to a large size, thus being able to greater prove the scalability of the algorithms. Also a great deal of work could be carried out into evolutionary algorithms with the agents. Since the agents have a relatively low level of complexity it would be relatively easy to program an evolutionary algorithm with a set of goals and see if the agents can achieve those goals. Research relating to particle swarm optimisation [11] could also be carried out.

Further research could also be carried out into more complicated behaviour's such as foraging. Due to the time limitations of the project foraging was not successfully implemented into the agents in time for the paper to be published. However the method for foraging has been laid out below.

7.1.1. Foraging

Foraging is another form of emergent behaviour and provides an efficient means to find quickly find a goal. Foraging involves finding a food source and translating the position of the food source to the other members of the group. This is demonstrated in ants with their pheromone trails. This will be accomplished with the agents, by the food source being a light. When an agent encounters this light source it finds the locations then follows the route back by utilising a linked list structure which contains all the changes in heading the robots made to get to that location. While the robots is moving to and from this location it will have it's own light source switched on signalling that it has found a food source. All the other agents in the range will then follow this robot until that food source has been reached effectively foraging and gathering the food.

8. Conclusion

The aim of this paper has been to investigate the properties of creating collaborative behaviours in multi-agent systems without the need for direct communication. This work in the combination of both investigations into communication and the level required to produce suitable results as well as an investigation into the behaviours that relatively simple cognitive mobile robots can achieve. In this paper we have proposed a methodology by which simple mobile robots can produce advanced forms of collaboration, by using a method of basic behaviours and little or no indirect communication.

This research has identified that each agent successfully exhibited group behaviour. However due to the lack of agents available we can only predict how the agents would scale. We do this using a simulation incorporating all the major information learnt throughout this paper. The simulation shows that the routines that have been developed are scalable up to 5 agents. Thus we can conclude that like natural flocks in theory there is no upper boundary to the number of agents there are in a flock, it is only limited by the range of each agent.

The methods of inter-robot localisation are fast and efficient enabling quick response times and fast moving robots. As opposed to a planner system where due to the delay caused by communicating and planning the robots would move slower, and be more susceptible to changes in the environment. This method mimics how natural flocks work without the need for communication. However the floor of the system was the constraint that the environment had to be dark. The robots were constrained within a dark environment due to the phototransistors being subject to a high level of natural light noise.

This paper is an extension to the study of natural intelligence specifically within multi-agent systems and their applications in the real world. The research shows that effective and advanced forms of collaboration can be accomplished without the need for communication or robots with a high cognitive level. Future work should lead to the reinforcement of the ideas set out in this paper and to broaden the understanding of multi-agent systems simulating biological behaviours.

9. References

- [1] Maja J Matarić, “*Interaction and Intelligent Behaviour*”, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, PhD Thesis, May 1994.
- [2] J. Kennedy, and R. Eberhart, “*Particle swarm optimization*”, in Proc. of the IEEE Int. Conf. on Neural Networks, pp. 1942–1948, 1995.
- [3] Reynolds, C. W. (1987) “*Flocks, Herds, and Schools: A Distributed Behavioural Model, in Computer Graphics*”, 21(4) (SIGGRAPH '87 Conference Proceedings) pages 25-34.
- [4] Holly Yanco, Lynn Andrea Stein, “*An Adaptive Communication Protocol for Cooperating Mobile Robots*”, Artificial Intelligence Laboratory Massachusetts Institute of Technology, 1993.
- [5] Dudek, G. Jenkin, M. Milius, E. Wilkes, D. “*A Taxonomy of Swarm Robotics*”, International Conference on Intelligent Robots and Systems, July 1993, pages 441-447.
- [6] Lawrence Withers, SE2P4 Lecture Notes, “*Mobile Robots*”, The University of Reading, January 2006.
- [7] Microchip Ltd. “*PIC16F631/677/685/687/689/690 Datasheet*” page 6.

- [8] Maja J Mataric', "*Integration of Representation into Goal-Driven Behaviour-Based Robots*", in IEEE Transactions on Robotics and Automation, 8(3), Jun 1992, 304-312.
- [9] B. Danette Allen, Gardy Bishop, Grey Welch, "*Tracking: Beyond 15 Minutes of Thought*", University of North Carolina, SIGGRAPH 2001.
- [10] Greg Wlech, University of North Caroline, Eric Foxlin, InterSense, "*Motion Tracking: No Silver Bullet, but a respectable Arsenal*", November/December 2002.
- [11] J. Kennedy, and R. Eberhart, "*Particle swarm optimization*", in Proc. of the IEEE Int. Conf. on Neural Networks, Piscataway, NJ, pp. 1942–1948, 1995.
- [12] M. Clerc, and J. Kennedy, "*The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space*", IEEE Transactions on Evolutionary Computation, 2002, 6, 58-73.
- [13] I D Kelly, D A Keating, "*Flocking by the fusion of sonar and active infrared sensors on physical autonomous mobile robots*". The Third Int. Conf. on Mechatronics and Machine Vision in Practice. 1996, Guimaraes, Portugal. Vol. 1, pp 1/1 - 1/4. (1996).
- [14] Berna-Koes, M, Nourbakhsh, I, Sycara, K, "*Communication Efficiency in Multi-Agent Systems*", Robotics Inst., Carnegie Mellon University, Robotics and Automation, 2004. Proceedings. ICRA '04 2004, pages 2129 – 2134.
- [15] Enee, G, Escazut, C, "*Evolution of communication in a gentic based multi-agent system*" Lab. I3S, Sophia-Antipolis, France, Congress on Evolutionary Computation, June 2004, pages 2038 – 2044.
- [16] Olfati-saber, R, "*Flocking for multi-agent dynamic systems: algorithms and theory*", Thayer Sch. Of Eng., Dartmouth College, IEEE Transactions on Automatic Control, March 2006, pages 401 – 420.

10. Appendix

10.1. Code Listing

```
# include <pic.h>
# include "stdbool.h"
# include <stdio.h>

// Configuration
__CONFIG(INTIO & WDTDIS & PWRTEN & MCLRDIS & UNPROTECT \
& UNPROTECT & BORDIS & IESODIS & FCMDIS);

// Declare global timers
volatile int time_s = 0;
volatile int time_ms = 0;
volatile int time_s_count = 0;

int PORTC_shadow = 0;
int PORTA_shadow = 0;
int PORTB_shadow = 0;

// Declare PWM variables
static int motor1_pwm = 0;
static int motor2_pwm = 0;
static int pwm_count = 0;

// Interrupt handler
interrupt void interrupt_handler(void)
{
    if(TOIF)
```

```

{
    ++time_ms;
    // Reset timer when gets to 977 (977Hz)
    if(++time_s_count == 977)
    {
        time_s_count = 0;
        ++time_s;
    }

    // move to next PWM phase
    ++pwm_count;
    pwm_count &= 0x7;

    // at start phase, clear motors
    if(!pwm_count)
    {
        PORTB_shadow &= ~0x80;
        PORTC_shadow &= ~0x38;
    }

    // is motor1_pwm negative?
    if(motor1_pwm & 0x80) {
        if(pwm_count >= - motor1_pwm) PORTC_shadow &= ~0x10;
        else PORTC_shadow |= 0x10;
    } else {
        if(pwm_count >= motor1_pwm) PORTC_shadow &= ~0x20;
        else PORTC_shadow |= 0x20;
    }

    // is motor2_pwm negative?
    if(motor2_pwm & 0x80) {
        if(pwm_count >= - motor2_pwm) PORTC_shadow &= ~0x8;
        else PORTC_shadow |= 0x8;
    } else {
        if(pwm_count >= motor2_pwm) PORTB_shadow &= ~0x80;
        else PORTB_shadow |= 0x80;
    }

    // write shadow registers to real
    PORTC = PORTC_shadow;
    PORTB = PORTB_shadow;

    TOIF = 0;
}

#define PORTA_mask(_AND, _OR) do { \
    PORTA_shadow &= _AND; \
    PORTA_shadow |= _OR; \
    PORTA = PORTA_shadow; \
}while(0)

#define PORTB_toggle(_bit) do {\
    PORTB_shadow ^= (1 << _bit); \
    PORTB = PORTB_shadow; \
}while(0)

#define PORTA_toggle(_bit) do {\
    PORTA_shadow ^= (1 << _bit); \
    PORTA = PORTA_shadow; \
}while(0)

static int count = 0;

// Routines for Ultra-Sonics
static void _ping_sonar1(void)
{
    // loop 8 times
    count = 8;
    GIE = 0;

    // we use assembler in order to get exact timing without using another timer
    #asm
    ping_sonar1_startloop:
    movf _PORTC_shadow, W

```



```

// now ping sonar and wait
ping_sonar_aux(which);
old_ms = time_ms;

//wait for comparator to fire (50ms timeout)
while((old_ms + 50 > time_ms) && !C1IF);

// compute range
range = (C1IF ? time_ms - old_ms : 0xFF);
if(which) sonar2_range = range;
else sonar1_range = range;
}

// sonar task
static int _sonar_task_next = 244; // 977/4
static bool _sonar_task_which;
static void _sonar_task(void)
{
    if(time_ms < _sonar_task_next) return;
    _sonar_task_next += 244; // 977/4

    ping_sonar(_sonar_task_which);
    _sonar_task_which = !_sonar_task_which;
}

void left_motor(int speed)
{
    motor1_pwm = speed;
}

void right_motor(int speed)
{
    motor2_pwm = speed;
}

static int trans_old;
static void trans_task(void)
{
    if (time_s == trans_old) { return; }
    trans_old = time_s;
    PORTB_toggle(6);
}

static int led_old;
static void led_task(void)
{
    if (time_s == led_old) { return; }
    led_old = time_s;
    PORTA_toggle(4);
}

// Behaviours
bool dummy = false;
int left = 0;
int right = 0;
int left2 = 0;
int right2 = 0;
bool leader = false;
bool lights = false;
int i = 0;

static bool Another(void)
{
    for (i = 0; i < 500; i++);
    if ((RB4 == 1) || (RB5 == 1) || (RC2 == 1) || (RA3 == 1)) {leader = false; dummy = true;}
    else { leader = true; dummy = false; }
    return dummy;
}

static void Explore(void)
{
    if (sonar1_range < 10) {left = time_s;}
    else if (sonar2_range < 6) {right = time_s;}
}

```

```

    else {left_motor(7); right_motor(7);}

    if (left+3 > time_s) {left_motor(2); right_motor(8);}
    else if (right+3 > time_s) {left_motor(8); right_motor(2);}
}

static void Home(void)
{
    if (RB4 == 1) {left_motor(6); right_motor(6);}
    else if (RC2 == 1) {left2 = time_s;}
    else if ((RB5 == 1) || (RA3 == 1)){right2 = time_s;}
    else Explore();

    if (left2+1 > time_s) {left_motor(-8); right_motor(8);}
    else if (right2+1 > time_s) {left_motor(8); right_motor(-8);}
}

static void Follow(void)
{
    if (sonar1_range < 7) { left_motor(8); right_motor(-8); }
    else if (sonar2_range < 4) { left_motor(-8); right_motor(8); }
    else Home();
}

int counter = 0;
static Flock(void)
{
    if (leader == true) { lights = true; Explore();}
    else {lights = false; Follow();}
}

long counter2 = 40000;
int i;
bool startup = false;

// Main
void main(void)
{
    // Set up OSCCON (Oscillator) to 8Mhz
    OSCCON |= 0x70;

    CM1CON0 = 0x80;

    //Disable all weak pull ups
    OPTION = 0b10000000;
    T0IF = 0;
    T0IE = 1;

    // Disable ANSEL and ANSELH registers
    ANSEL = 0;
    ANSELH = 0;

    // Pin setup
    TRISA = 0b00001111;
    PORTA = 0b00001111;

    TRISB = 0b00110000;
    PORTB = 0b00110000;

    TRISC = 0x4;
    PORTC = 0x4;

    // Disable weak pullups
    WPUB = 0b00000000;
    IOCB = 0b00000000;

    // enable interrupts
    GIE = 1;
    PEIE = 1;

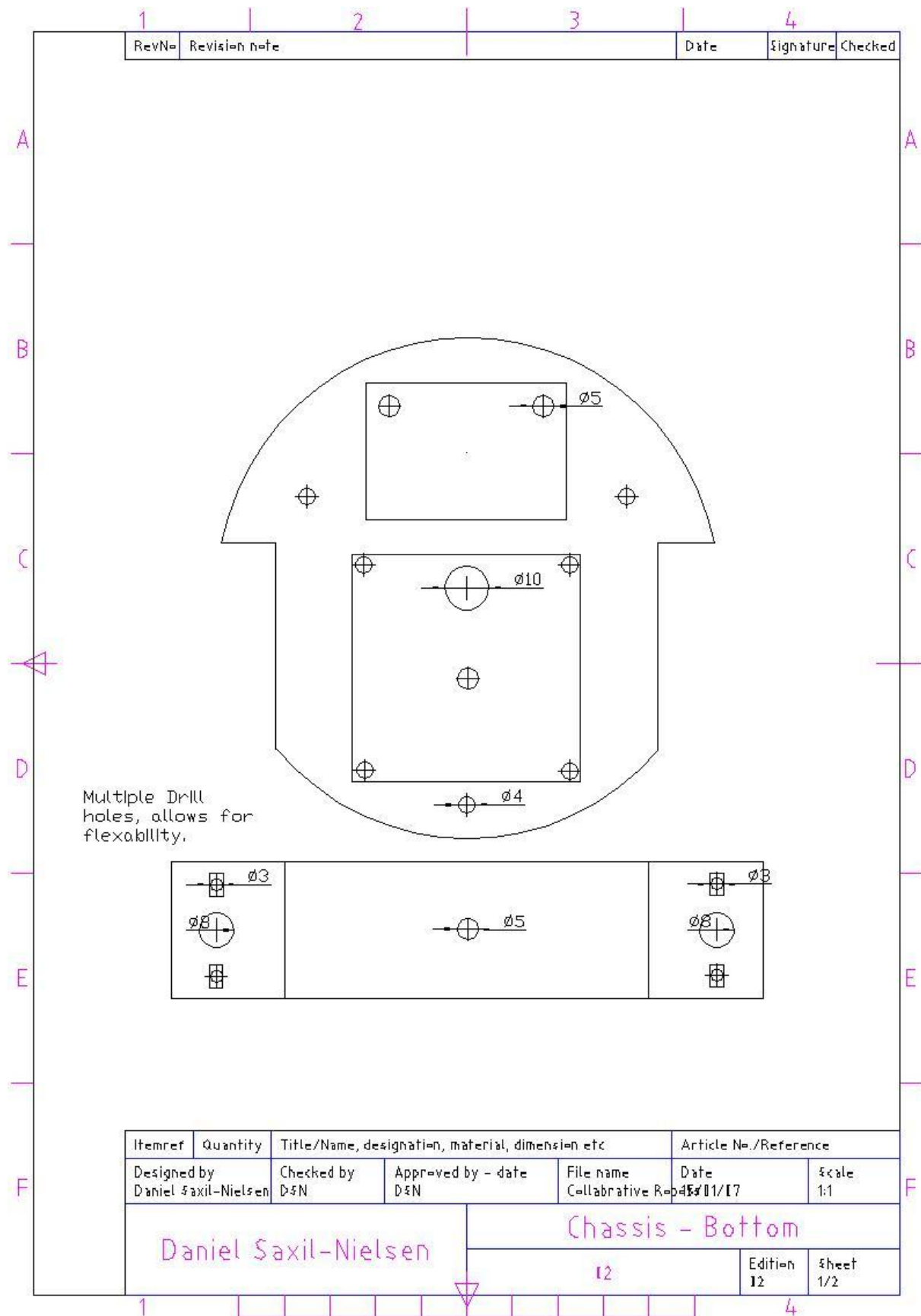
    while(1)
    {
        // start tasks
        _sonar_task();
    }
}

```

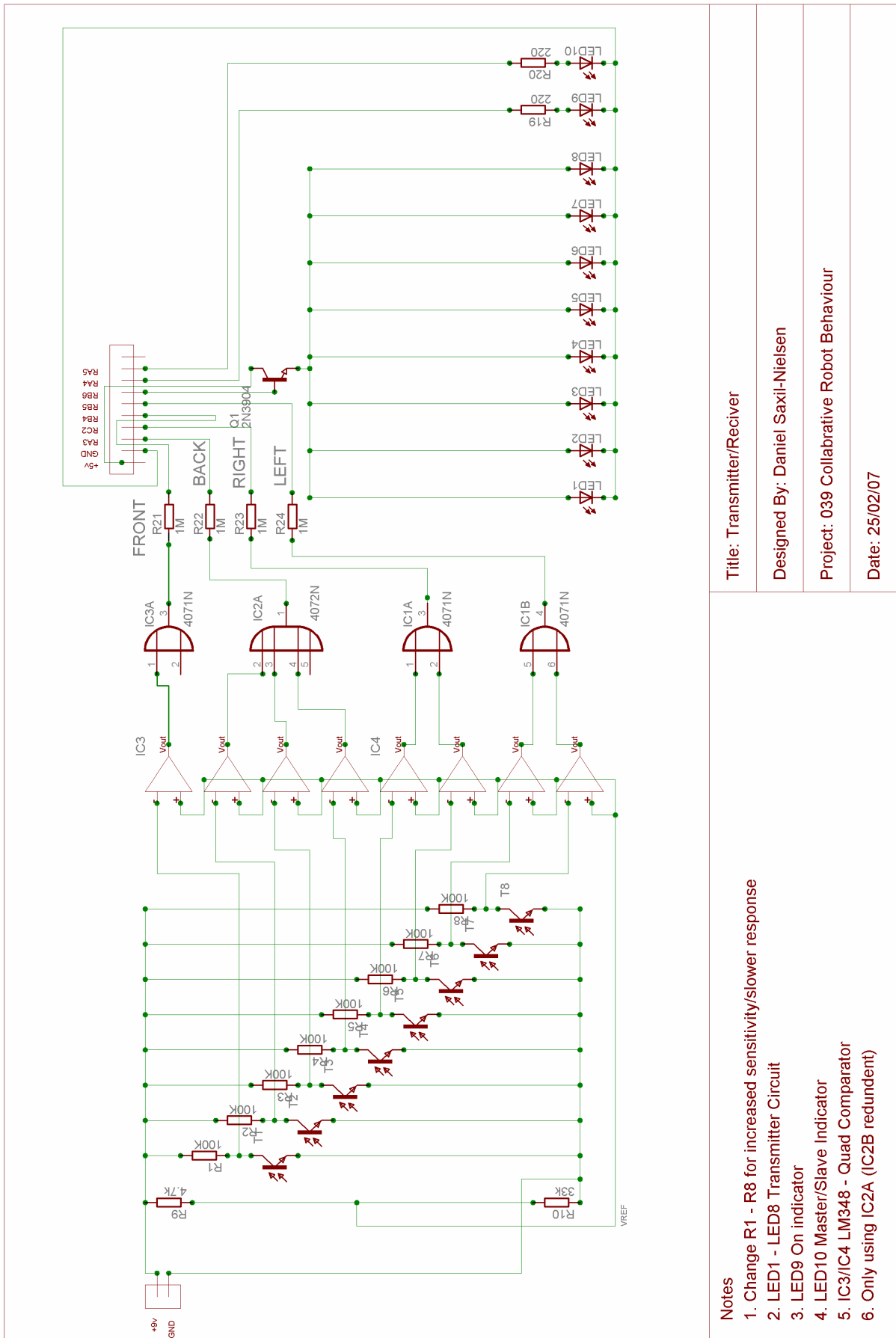
```
if (leader) { RA4 = 1; RA5 = 0; } else { RA4 = 0; RA5 = 1; }
if (lights) RB6 = 1; else RB6 = 0;

if ((leader) && (counter2 == 10000)) //1 second
{
    if (lights)
    {
        RB6 = 0;
        if (Another()) leader = false; else leader = true;
        RB6 = 1;
    }
    else { if (Another()) leader = false; else leader = true;}
    counter2 = 0;
}
else
{
    if (counter2 == 50000) //5 seconds
    {
        if (lights)
        {
            RB6 = 0;
            if (Another()) leader = false; else leader = true;
            RB6 = 1;
        }
        else { if (Another()) leader = false; else leader = true;}
        counter2 = 0;
    }
}
counter2++;
Flock();
}
```

10.2. Chassis Diagrams

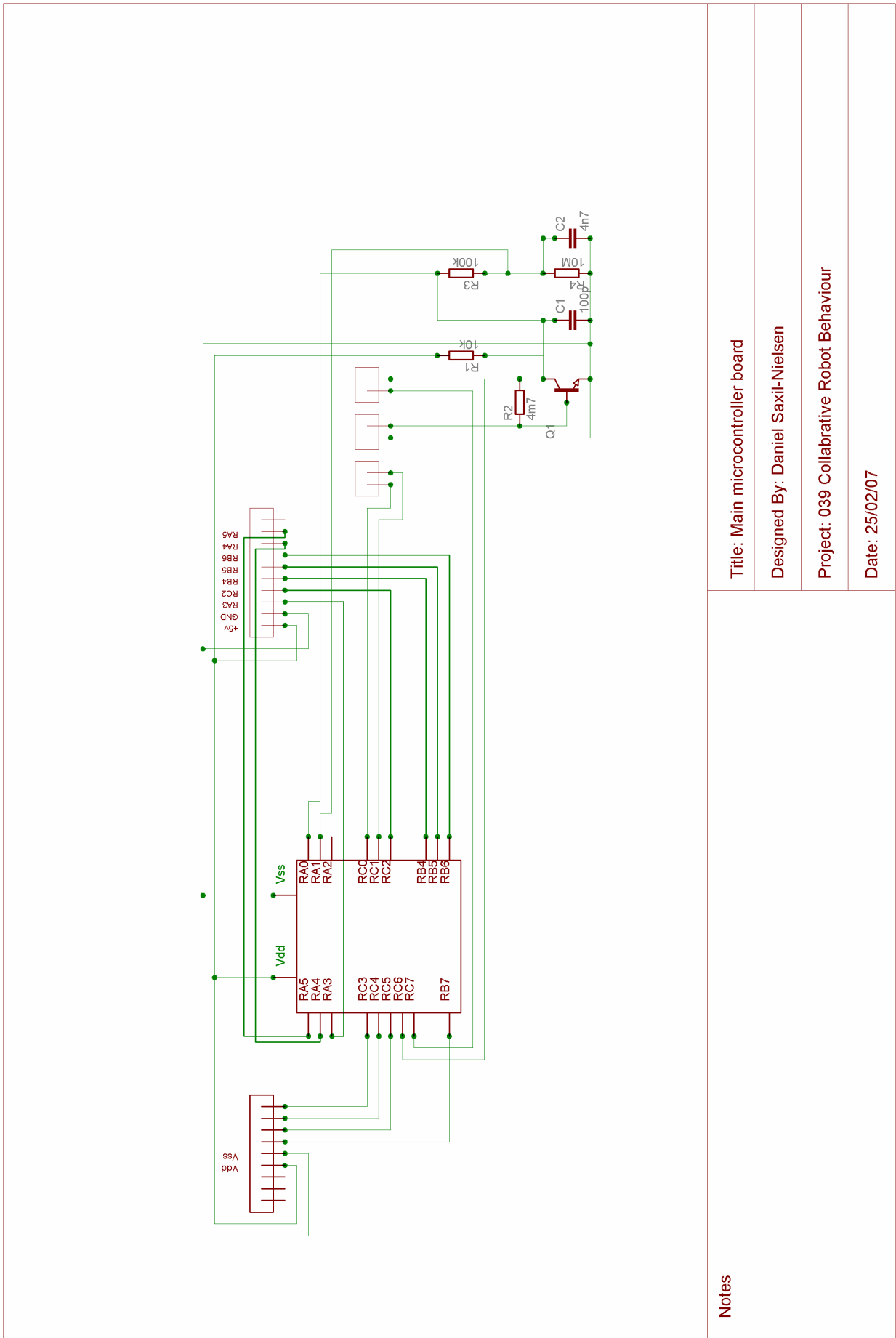


10.3. Circuit Diagrams



Title: Transmitter/Receiver
Designed By: Daniel Saxil-Nielsen
Project: 039 Collaborative Robot Behaviour
Date: 25/02/07

- Notes**
1. Change R1 - R8 for increased sensitivity/slower response
 2. LED1 - LED8 Transmitter Circuit
 3. LED9 On indicator
 4. LED10 Master/Slave Indicator
 5. IC3/IC4 LM348 - Quad Comparator
 6. Only using IC2A (IC2B redundant)



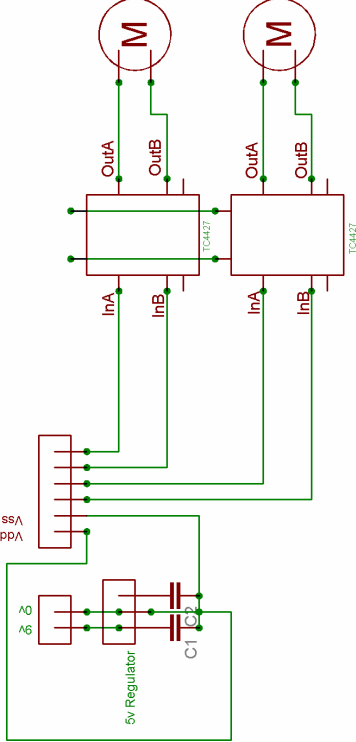
Notes

Title: Main microcontroller board

Designed By: Daniel Saxil-Nielsen

Project: 039 Collaborative Robot Behaviour

Date: 25/02/07

	<p>Notes</p>
	<p>Title: Motor Driver Board</p>
	<p>Designed By: Daniel Saxil-Nielsen</p>
	<p>Project: 039 Collaborative Robot Behaviour</p>
	<p>Date: 25/02/07</p>

10.4. PCB layouts

