Imperial College of Science Technology and Medicine
Department of Electrical and Electronic Engineering
1st Year Electronics Laboratory
EEBUG Group Design Project

Report cover sheet

Please complete this cover page.
Save it to your PC, and then insert it as front page of your report.
Print the complete report as a pdf file, and check that the formatting has not been affected.
Finally, the **group secretary** must upload the pdf to Blackboard
(EE1-LABE EEE 1st Year Electronics Lab (2015-2016)/ Group Design Project assessments)

Tutor	Dr I M Jaimoukha
Design Group *(tutor's initials, followed by group number)*	IMJ - 2A
Report type	Stage 2
Submission date	Wednesday, 23 March 2016

Checklist (see labweb for guidance)		Yes / No
Is the document within the word limit?	Word count 7925 (max 8000) No of figures 11 (Max 12)	☒ / ☐
Are all figures and graphs clear and complete?		☒ / ☐
Have you made a full list of references?		☒ / ☐
Have you included all relevant diagrams?		☒ / ☐
(Stage 2 only) Have you included details of your component orders?		☒ / ☐
Have you read and understood the college plagiarism statement ?		☒ / ☐

Plagiarism statement *"I certify that this report is our own original work, and that any other sources are fully acknowledged"*	*Group leader logon (eg xyz15) zw4215*

LATE SUBMISSION *The Department's publicised policy is that coursework submitted after the deadline automatically fails.* *However, if there are genuine extenuating circumstances which make it impossible to submit on time, please explain.*	*Reasons for late submission:*

Contents

1 Introduction

In this term the team is required to complete the design of the EEBug, and build a prototype for it. To accomplish this task, a digital approach centred on a microcontroller is used. Multiple options are considered and compared against each other. This report describes the process of this design, before explaining the technical details.

2 Project Management

2.1 EEBug Design Specifications

2.1.1 Requirements

The main requirements of the device are set by the client as described:

- Follows a curved line of decreasing blackness drawn on a white board.
- At the end of the line, it has to continue straight for 10cm.
- Then draw a circle with decreasing radius, completing at least one full rotation before stopping.

2.1.2 Objectives

The objectives of the project are defined as below:

2.1.2.1 Stay on Budget

The budget is set at £8 per bug, per device. Since it is not a very high budget, proper management needs to be carried out, and the budget is tracked every step of design process.

2.1.2.2 Deadline

Deadline for the design is set the 23rd of March. So until then, appropriate workload distribution should be done among team members to make overall project more manageable and enable the team to reach the deadline without any problems.

2.1.2.3 Compliance of the restrictions of project

Limitations and requirements are defined by the client. The final product should satisfy all these even if Superbug functions are to be implemented.

2.1.3 Restrictions

Some restrictions are also stated by the client. Those are taken into consideration and a checklist is done to show the completion of each one.

Restriction	Completion	Notes
Maximum Number of Sensors:2	X	2 infrared sensors used
No additional mounted boards	X	
Maximum of 4 AA batteries	X	
Completely autonomous: no human intervention after turning on.	X	All actions autonomous after switching on
Basic bug architecture should be kept constant.	X	Fuses, breadboard, motor used are the same as the original bug
Number of Microprocessor pins less than 8	X	ATtiny used: 8 pins
If the bug reaches the black border, it must stop.	X	
The bug must not be confused by uneven lighting conditions	X	Sensors are chosen in order to perform this task
Must have a proper switch to start	X	
Logbook must include single page of instructions	X	
Superbug Function	X	Described in section 5.1.
Budget Available: £8	X	Described in section 3.3

2.1.4 Deliverables

Deliverable Title	Term	Week
~~Client Meeting Unofficial Report~~	~~Autumn~~	~~10~~
~~Management Report~~		~~11~~
Video “Pimp my bug”	Spring	8
Design Report Part 1		10
Design Report Part 2		10
Viva	Summer	9

The deliverables required this term is an individual video which advertise the EEBug as a toy for and to submit a report which consists of two parts, Engineering Design Process and Technical Outcomes.

2.2 Progress since last report submission

Since the last report, significant progress has been made in almost all aspects of the product. This is especially true in the area of its design and functionality.

During Christmas break, team members worked individually looking for ideas of how to implement the EEBUG following the requirements and restrictions stated by the client.

A meeting is then scheduled during the first week of the term in order to decide which of the three approaches we are going to construct for the group EEBUG. The other two ideas are also discussed and may be used as backup solution if the chosen approach fails.

Feedback from first report is taken into serious consideration and improvements on the group management and enhancements description is done in a more realistic and detailed way.

In general, appropriate work was carried out in order to fulfil the requirements.

2.3 Project Planning

2.3.1 Aims

The aim of this term is to implement a solution that meets all requirements and restrictions stated by the client. Objectives of the project are as described before. Appropriate organisation should be done in order to accomplish each of them.

To stay in budget, project costs should be tracked to avoid over budgeting situations. To satisfy the deadline without problems, appropriate milestones should be set so that overall project will be more manageable. And finally, to stay in the project's restrictions, a careful study of the requirements should be done in combination with the equally distribution of the workload among team members.

2.3.2 Work Packages

A better way to manage the task stated before is to split up the work load into several, more manageable sections that later on will form the finalised solution. This can be visualised using a tree structure diagram (top-down design method).

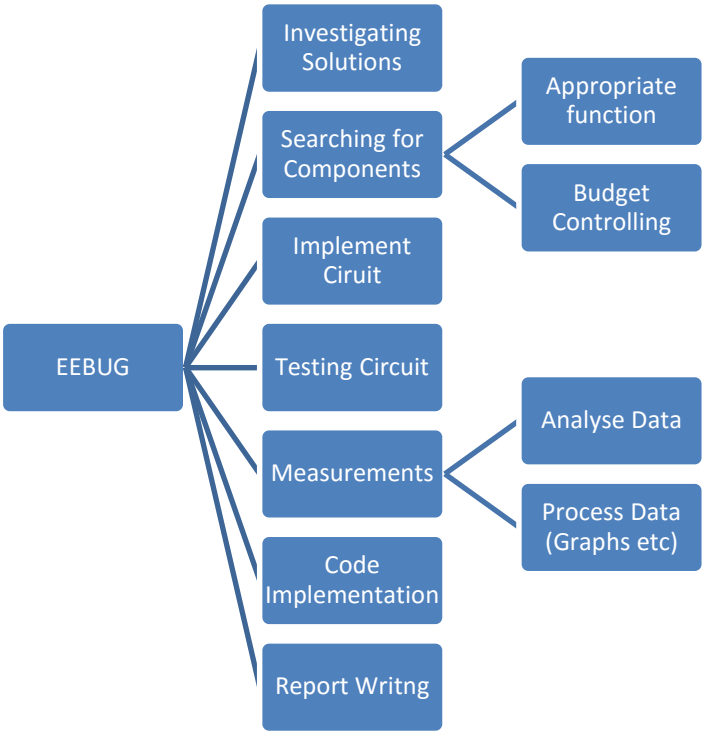

Figure 1 Work Packages

2.3.3 Allocation of Work packages

Later on, work packages need to be assigned to team members equally in order to speed up the time of the completion of the final product, as well as specialising the research areas of the team members in particular sections, so that later on we can teach each other. The work packages are distributed as follows:

Work Packages		**Team Member(s) Responsible**
Investigating Solutions		All Team Members
Searching for Components	Appropriate Function	Wang Zifan
	Budget Controlling	Alejandro Gilson
Implement Circuit		Wang Zifan Loizos Papachristoforou
Testing Circuit		Alejandro Gilson
Measurements	Analyse Data	Wang Zifan
	Process Data (e.g. Graphs)	Loizos Papachristoforou
Code Implementation		Loizos Papachristoforou Alejandro Gilson
Report Writing		All Team Members

2.3.4 Milestones - Project Plan

As it can be seen in the Figure 2, the work packages are organized chronologically.

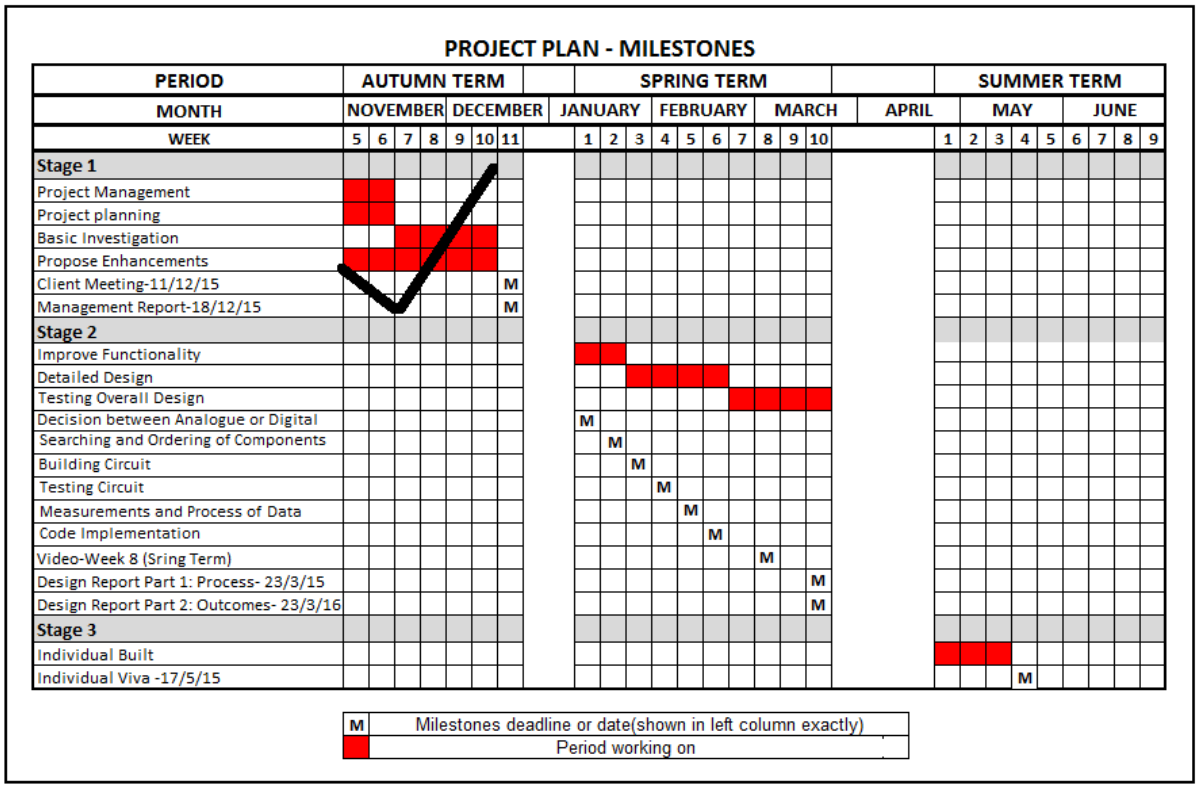

Figure 2 Milestones

During the first two weeks of the term the different approaches that were suggested were discussed to decide on which to follow. Appropriate components for the chosen approach were then selected and ordered.

The next four weeks, the implementation of the design needs to be done. This is comprised of the building and testing of circuit, taking measurements and processing of data as well as the implementation of the code for the digital approach.

The last four weeks of term, the overall design has to be tested and improved in order to fulfil the requirements.

2.3.5 Meetings

The same system of meetings is used like previous term. Every Thursday at 17:00 a meeting is set up in the Electronics Laboratory or Computing Laboratory, due to the convenience of timetables and availability of resources. However, if a deadline approaches, additional meetings will be organized in order to accomplish every task.

2.3.6 Personnel Issues- Red Flag Issues

Like previous term, no red flag problem has interfered with the project development.

In case it was to happen the same procedure will be followed. First, the person has to be warned about the situation by the other members. If after sometime the problem remains then action will be taken, and the group tutor, Dr. Jaimoukha will be informed.

2.4 Proposals for future Work

Several improvements have been studied in order to achieve a more perfect design that could work better and make the Bug more attractive to the user.

One of them is to make the Bug follow a line of any colour on any kind of surface (not just white). However, as it is obvious, the colour of the line and the surface cannot be the same. After some time brainstorming the team ended up with the idea of calibration.

Another idea is the implementation of an array of LEDs to make more elegant and entertaining the device.

2.4.1 Calibration

A huge number of factors, such as having a different surface material, could affect the detection of light from the Bug. In order to avoid this, a process of calibration that reads the voltages of the line is required. Not only would this eliminate noise but would it also allow the Bug to work on many other surfaces and lines with different colours.

It would work as follows: The user puts the Bug on the centre of the line and switches on the device. For a few seconds the Bug will just read the voltages of the line and save the values as a reference. After this the Bug will start its movements using the measured values instead of presets.

2.4.2 Use of LEDs

- The LEDs can make the Bug more informative because it could tell the user at which stage the Bug is. A possibility studied by the team was to switch on three separate LED with different colours to indicate three different stages.
- Another implementation of the LEDs is to tell in which direction the Bug is moving. In this way if the Bug was moving to the right, an LED situated at the right side of the device would turn on and vice-versa.
- A final idea for the LEDs is to show the speed at which each motor is running by varying its brightness or the number of LEDs, this would tell if the Bug is turning and would have a nice effect while performing the spiral.

All of these ideas could be implemented at the same time with the use of an array of LEDs. They can be simply controlled by the microcontroller, but if the number of LEDs is too big a multiplexer could be used.

3 High Level Design

The lack of strict requirements for the construction of the Bug allows a wide range of design options. This means that there are many ways of building this gadget. However, as most electronic designs, they can all be categorised into two general approaches: Digital and analogue.

3.1 Proposed Solutions

As mentioned before, three ways of implementing the prototype EEBug are considered. Two of them uses digital approach while the other one is implemented using an analogue approach.

A brief description of each method is included in this part of the report with their respective advantages and disadvantages.

3.1.1 Analogue solution

An analogue approach provides a complex yet elegant solution for the Bug. It involves the use of several op-amps and logic gates, organised according to specific stages of the journey: Following the line, 10cm and the final spiral.

Our team explored this solution at an early stage of development. The basic circuit that we designed was based on a differential amplifier and a comparator. This kind of circuit evaluates the difference between two input voltages given by the sensors. If one of the sensors saw a different shade it would generate a higher voltage and the op-amp would amplify the difference in voltages between the two terminals. Afterwards in the second stage, the circuit would check if it's between some boundary voltage set up by some resistors. This was just a basic idea and in order to accomplish the goal further designs had to be made.

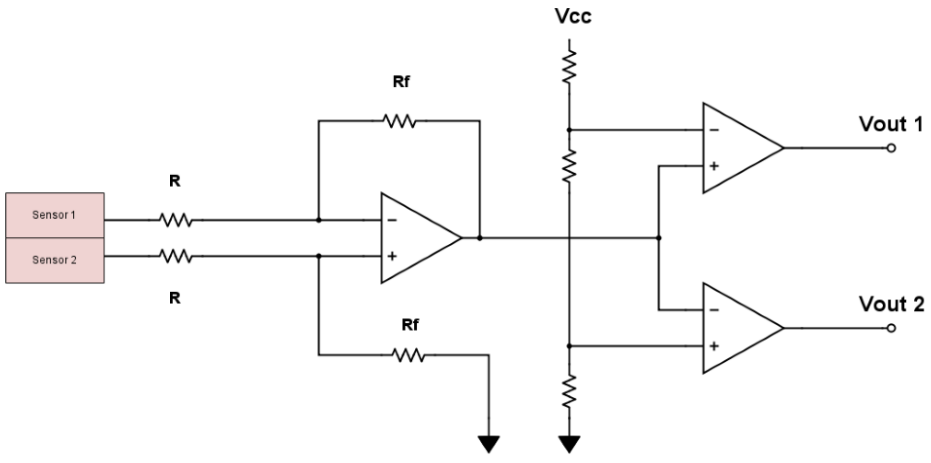

Figure 3 Analogue approach circuit. Generated using: http://www.digikey.com/schemeit/ [Accessed 16/03/16]

Advantages and disadvantages

As mentioned above, this design is elegant as the voltages are continuously read and therefore the reaction of the circuit is almost instantaneous.

However, such an implementation can be extremely complex, as it requires a high level of circuit design knowledge. The construction of such a circuit demands a disproportionate amount of testing at every stage. Moreover It gets difficult to debug as the circuit grows.

3.1.2 Digital solutions

Digital processing of information can handle extremely tedious operations with ease with a programmable microcontroller. The chip reads the voltages coming out from the sensors, processes it and generates a digital output. Due to the fact that a digital device can only provide in most of the cases two voltages, in order to apply an intermediate output a PWM has to be used.

Pulse Width Modulation (PWM)

PWM provides a way for microcontrollers to give a good approximation to real analogue outputs. This is done by varying the duty cycle, or the percentage time that the signal stays on. At sufficiently large frequency the device would behave as if the output was its average. For example, an output of 5V at 50% duty cycle is the same as giving the bulb a 2.5V DC supply. Thus by modulating the pulse width, any desirable value within 0 to 5V can be achieved.

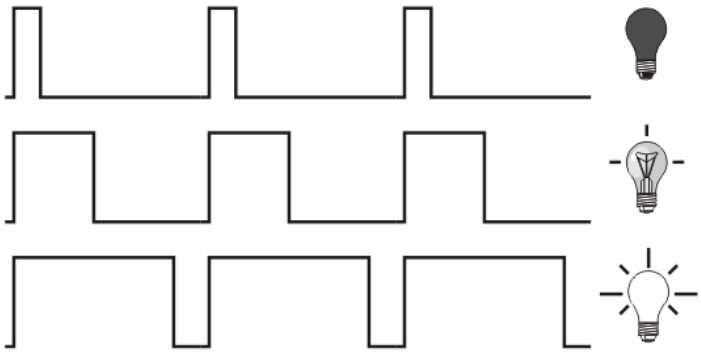

Figure 4 Pulse width modulation. Available at: http://learn.mikroe.com/ebooks/picmicrocontrollersprogramminginassembly/chapter/ccp-modules/ [Accesed 18/03/16]

3.1.2.1 Arduino programming

Arduino Uno is a microcontroller that uses a programing language very similar to C++, which the team is very familiar with. The controller also has many analogue inputs and PWM outputs for programming use.

Due to the restrictions of the project, an ATtiny has to be used to shrink the relatively huge Arduino board. The ATtiny is a microcontroller used in digital circuits. It has analogue inputs

and PWM outputs just sufficient for the EEBug. This device can then be programmed using Arduino as an in-system programmer (ISP), thus allowing the use of its coding language.

Advantages and disadvantages

Arduino is easy to use and is an inexpensive platform. The language it uses is known by the team members and therefore not much additional code study is required. However, because it is a high level programing language it is not as efficient as it would be with other coding languages such as Assembly.

3.1.2.2 PIC Microprocessor programming

The peripheral interface controller (PIC) is a digital device that can be programmed using a computer through a PICkit. The programing code the team intended to use was Assembly. This a very low level programming code that allows the creator to write very efficient programs and be in a close touch with the hardware that is used.

Advantages and disadvantages

PIC Microprocessor programing is very good in terms of efficiency due to its nature. However, this requires much higher understanding of hardware and its language. As the project has a tight deadline, this is arguably ends up being more of a disadvantage.

3.1.3 Choice of Sensors

The information that has to be given to the Bug has to do with its position on the board. It could be on a white surface, on the black line it has to follow or it could be on the black thick borderline. This is visual information and therefore it has to be obtained accordingly. It is for this reason that the Bug needs sensors which to read this information. During the first stages of design the team came up with two options: Light sensors in the visible and infrared frequency ranges.

3.1.3.1 Visible light sensors

Brief description

These kind of sensors read the amount of light in their environments. They act accrodingly to the brightness they receive. Inside this group two different types of photo-sensors were studied and mentioned in the previous report: Photodiodes and Light Dependent Resistors (LDRs).

Light Dependent Resistors

As the name suggests, this sensors has a light varying resistance. The larger the brightness seen by the LDR, the smaller the resistance it will have and therefore, more current can flow.

Photo-diodes

These components are basically diodes that provides current as a function of brightness. By connecting the diode in reverse bias, no current would flow through normally. When light is shone onto it, a photoelectric current is generated which increases the reverse saturation current.

Implementation in the circuit

The idea for this approach was to insert two light sensors at the front of the Bug pointing down and at a suitable distance so that each sensor could read the border of the line. If one of the sensors saw more light reflected from the ground this would have meant that the Bug was too far in that direction.

Advantages and disadvantages

At a first stage of development the visible light sensors seemed very convenient. The team already had useful information about them from the previous investigation, they were relatively cheap and they did not need any kind of light emitter, as environmental light was more than enough.

Conversely, as development continued the team concluded that ensuring the bug was not affected by environmental light noise, caused by uneven lightning conditions of the room, would be very challenging.

3.1.3.2 Infrared sensors

What is infrared?

Infrared radiation is an electromagnetic wave with a wavelength longer than visible light, at approximately between 300GHz and 430 THz.

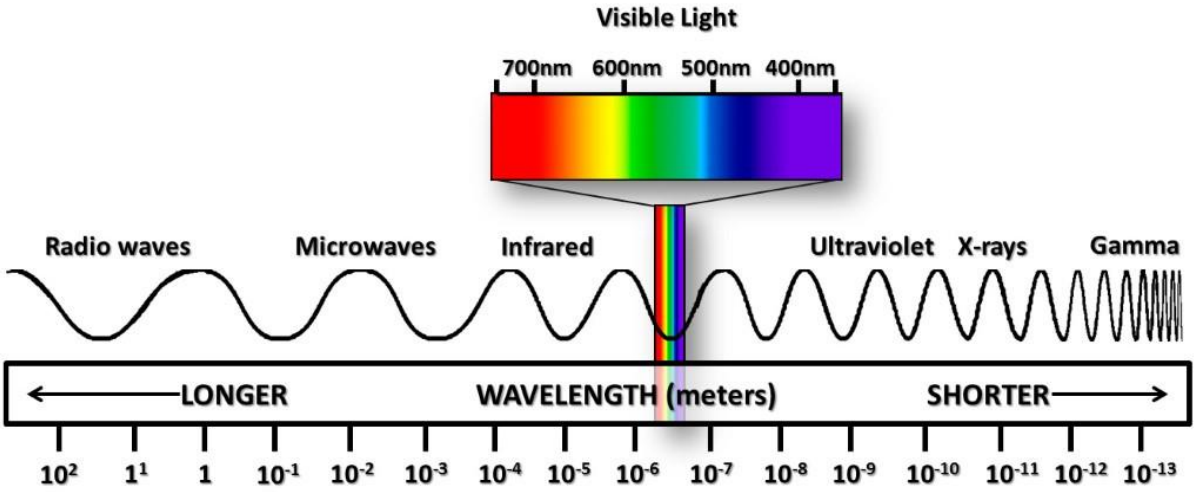

Figure 5 Electromagnetism radiation spectrum. Available at: http://masterdigitalcolor.com/2014/11/what-the-heck-is-a-gamut/ [Accessed 15/03/16]

Implementation in the circuit

The testing environment does not have perceptible infrared radiation for the Bug to read. It is because of this reason that an infrared emitter is preferred.

The emitter is located at the front of the Bug looking downwards, with two sensors reading the borders of the line on each side of the LED. A higher percentage of Infrared radiation would be from a white surface. Therefore sensors allow more current to go through, compared to a darker surface.

Advantages and disadvantages

The infrared approach eliminates any kind of noise interference with the environment. However, in order to make the design work an emitter is needed. This increases the cost of the Bug.

3.2 Final design

At the end the team decided to:

- Take a **digital approach** instead of an analogue one.
- Use an **Arduino** instead of a PIC as the programming device.
- Use **infrared sensors** instead of light sensors as the method getting appropriate input.

The team took this decision mainly in order to make the project as simple as possible. However, it would have been interesting to take the analogue approach. The following figure shows the decision matrix followed. Efficiency is related to the amount of processes the device has to take in order to realize a function.

Criteria	Analogue	Arduino	PIC	Infrared sensors	Photo-diodes	LDR	Importance
Power cons	3	8	8	9	5	5	7
Cost	5	9	9	3	8	8	4
Simplicity	3	9	3	5	7	7	8
Efficiency	9	5	7	9	5	5	7
Merit	9	6	8	6	4	4	6
Value	182	**235**	217	**214**	182	182	

Why to choose digital approach instead of analogue approach?

The main reason why the decision of following a digital approach is done is due to the safety of the solution. What is meant by this is that a digital approach will have a smaller percentage of failure due to its simplicity. Analogue solution might be more neat and impressive but it requires an in-depth knowledge of op-amps applications and analogue systems thus plenty of additional time to spend on extra research and experimental investigations of the solutions.

Why to choose Arduino instead PIC?

The two programming devices have different assembler, different memory architecture, different input and output capabilities and different tools. Arduino is preferred due to the fact that it can be programmed using C++ language which the team is much more familiar with. On the other hand PIC can be programmed using assembly language in which we have limited knowledge.

Why to choose infrared sensors instead of light sensors?

Sensors is very important to give proper input. In that way infrared are decided instead of light sensors due to the fact that they eliminate any kind of noise interference with the environment as they do not interfere with visible/environmental light. It may be more expensive in construction as an emitter is required but the cost is affordable for the available budget.

3.3 Cost of enhanced bug

For the accomplishment of this project the team is given £32 to spend in components. £8 for each member (3) and £8 more for prototyping. In addition, there are several restrictions in the ordering of components. There cannot be additional boards mounted on to the bug, surface-mounted components, the components have to come from specific suppliers, and they cannot be stocked in the US.

Total cost of the project

In order to achieve the goal of the project there was a budget constraint that had to be allowed for. In this way, the team cannot spend more than £8 on the design and no more than the same amount for each individual Bug with its further functionalities. It is because of this that the team had to be very careful in the order of the components and not breaking them. This however, did end up happening. Details can be found in section 4.6.2.

The total spent on the prototyping process was £6.59. The total for the final design costs is £5.85.

The following figure shows the components ordered and specifications.

Components	Max current	Max Voltg	Min voltg	Output	Size (mm)	Other	Price (£)	Quantity	Total (£)
C&K COMPONENTS OS202013MT5QN1 Slide Switch, OS Series	200mA	30V DC	_____	_____	11.6 x 9.10	Vertical orientation	0.36	1	0.36
MURATA PKM22EPPH2001-B0 SOUNDER	_____	30V (p-p)	3V	_____	22x22x10.9	70dB 2KHz Resnt Freq	1.03	1	1.03
TEXAS INSTRUMENTS LP2950ACZ-5.0 Fixed LDO Voltage Regulator	200mA	30V	5.4V	_____	_____	O/P 5V Drop out 380mV 3 pins	0.681	1	0.681
KINGBRIGHT L-57IID LED, Low Power, Red	160mA	2V	_____	_____	5	Wavelenght 625nm 20 mcd	0.16	1	0.16
OSRAM SFH203-FA PHOTODIODE, IR FILTERED	_____	_____	_____	_____	5.9	Max senst wavelnght 0.9µm	0.22	2	0.44
ATMEL ATTINY85-20PU 8 Bit Microcontroller ATtiny	_____	5.5	2.7	PWM	9.27 x 7.62	20MHz 8 pins	0.96	1	0.96
VISHAY TSAL6200 Infrared Emitter	200mA	_____	_____	1.35	5	17° angle	0.379	1	0.379
SD0190 TRANSISTORS BJT 2N3904 T092 NPN	200mA	_____	_____	_____	_____	NPN	0.15	2	0.3
KOA SPEER ELECTRONICS MFS1/4CC2003F 200 kohm	_____	500V	_____	_____	_____	5% tolerance	0.0553	2	0.1106
MULTICOMP MCF 2W 100R 100 ohm	_____	500V	_____	_____	_____	5% tolerance	0.219.	4	0.876
MULTICOMP MCF 0.25W 10K ohm	_____	500V	_____	_____	_____	5% tolerance	0.0384	1	0.0384
MULTICOMP MF25 1K ohm	_____	500V	_____	_____	_____	5% tolerance	0.0536	1	0.0536
MULTICOMP MCFYU6104Z6 Capacitor 0.1 µF	_____	50V	_____	_____	_____	80%/-20% tolerance	0.1143	4	0.4572

3.4 Rationale for the choice of components

Slide switch

One of the requirements for the project was to have a proper switch instead of using a wire to connect two points on the breadboard. It did not take long for the team to come up with the idea of buying the *OS202013MT5QN1 Slide Switch, OS Series, double pole-double throw (DPDT)*. This specific switch was chosen because its maximum current of 200mA was sufficient for the circuit, and it had a reasonable price. Although a single pole-single throw would be adequate, the DPDT version was cheaper, thus it was selected.

Piezo sounder

One of the extra functionalities that had been given to the bug was the capability of generating music during its movements. The sounder converts voltage signals into sound. *MURATA PKM22EPPH2001-B0 SOUNDER* was chosen due to a few reasons. Firstly, its resonant frequency is at 2 kHz compared to the 4 kHz that most of its counterparts have. It is also much more pleasant to the ear to have a tune around the sixth octave than the seventh. The voltage range that it supports is also well beyond the voltage that the microcontroller will provide.

Voltage regulator

A voltage regulator was necessary because the microcontroller requires a 5V voltage supply. 4 AA batteries in series can supply a maximum of around 6.4V, thus a low-drop out regulator is necessary to keep the power loss at minimum. Thus the 5V output voltage with the smallest dropout voltage was chosen, which is the *TEXAS INSTRUMENTS LP2950ACZ-5.0 Fixed LDO Voltage Regulator*. Implementation of this device will be explained later in further detail.

Photodiode

This device controls the current that flows through itself according to the IR radiation coming from the IR-Emitter. The main reason to choose *OSRAM SFH203-FA PHOTODIODE* was that its detection wavelength matches that of the IR-emitter, which is 750 to 1100 nm.

Microprocessor ATtiny

The ATMEL ATTINY85-20PU 8 Bit Microcontroller, ATtiny was chosen due to its high memory storage (512 KB), its pin configuration, and its compatibility with the Arduino. It also has 8 pins, which is the maximum allowed in the project specifications.[1]

IR-Emitter

TSAL6200 provides infrared light at a wavelength of 940 nm. This particular LED was chosen because its wavelength fits the range of the detector. It provides sufficient brightness for the photodiodes to respond to, at the voltage a few batteries can supply.

Transistor

SD0190 TRANSISTOR BJT 2N3904 T092 NPN transistor was chosen because it is readily available. It also supplies sufficient current for the motor.

4 Detailed Design

4.1 Enhancement realisation

4.1.1 Implementation

In order to achieve the task given by the client, the EEBug needs to follow a line as it fades to white, then travel straight for 10 cm, followed by drawing a spiral before coming to a stop.

To accomplish this, a LED is installed to shine infrared light on the line. Two infrared sensors are placed on either side to detect the reflected light, thus telling the bug its relative position. The ATtiny is programmed to take this information and send signals to the motors, which would determine the speed that each motor runs at. This then translates to movement of the bug.

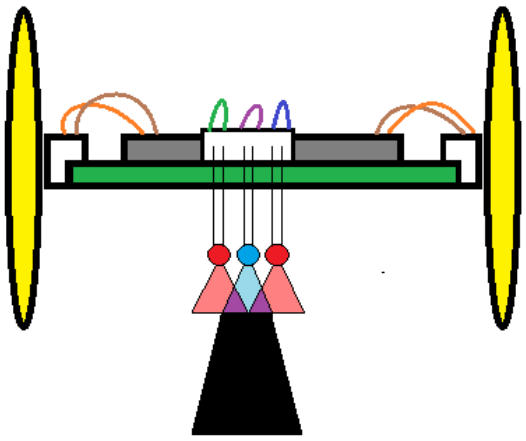

Figure 6 Design sketch

4.1.2 Problems may be encountered

- The reflected light range may not be sufficiently big enough for the ATtiny to get proper input and give the appropriate output.
- The two sensors may not be exactly identical. This will result in different input voltages to the pins of ATtiny when the absorbed reflected infrared light is the same. This will affect the output speed of the motors, as when, for example, both sensors absorb light reflected from the same colour surface they will rotate at different speeds.

4.2 Circuit – Diagram

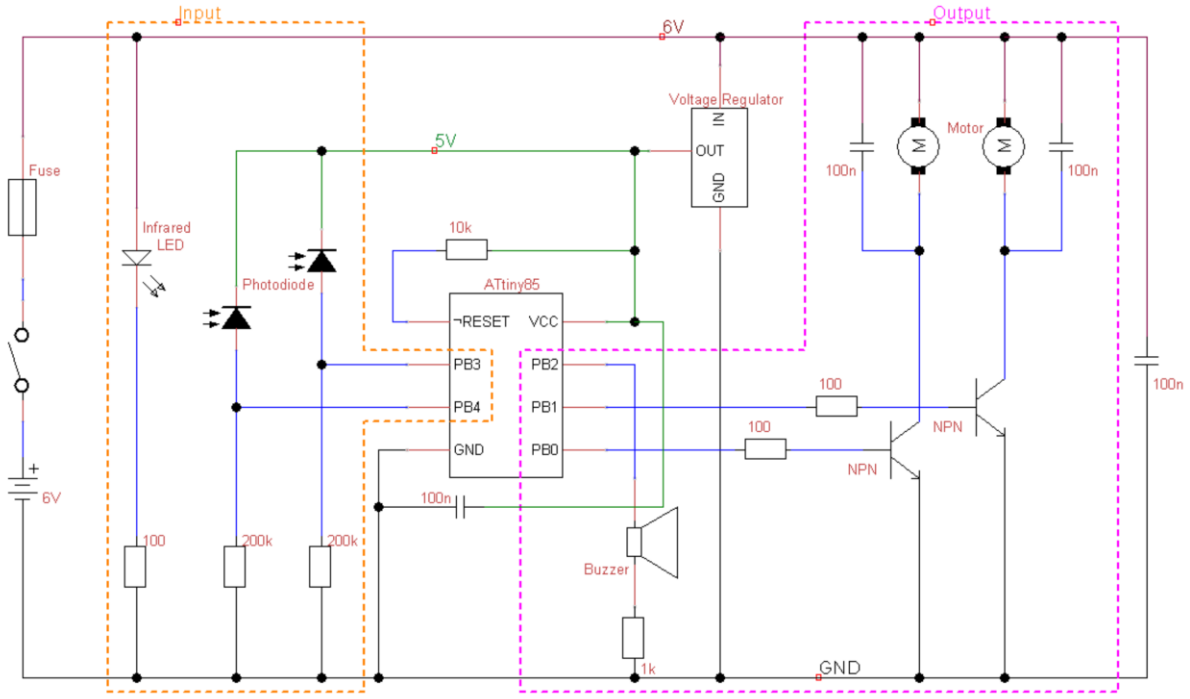

Figure 7 Circuit Diagram

4.3 Circuit – Summary

The centrepiece of the circuit is the ATtiny, a Microcontroller that functions as the brain of the bug. It is surrounded by the main circuit which supports its functions. The main circuit can be separated into an input and an output stage. The input stage collects information from sensors and sends them to the controller, and the output stage brings the processed signals to the motors and buzzer. Thus with the stages working together, the bug can interact with its environment.

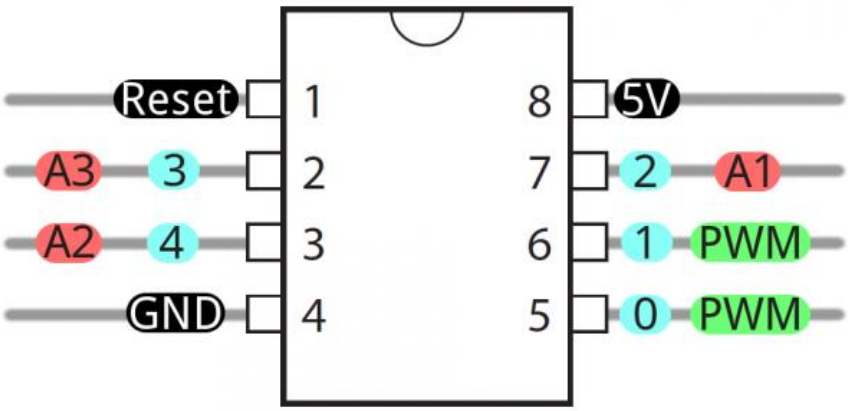

Figure 8 Pins of the ATtiny85[1]

4.3.1 Circuit – Input Stage

This stage consist of two infrared photodiodes and one infrared LED.

- Light Emitting Diode

The LED emits infrared light rated at 940 nm. It is connected across the 6V power rail to ground, while in series with a current limiting resistor for protection. This provides the infrared light required for the infrared photodiodes to function. The LED has an angle of half intensity of 17°. As such it is placed close together physically with the photodiodes in order to utilise this output with maximum efficiency.

The benefit of using light signals outside of the visible spectrum is that background lights are effectively filtered out, and thus will not affect the detection of the line. This means the bug would be invulnerable to most foreign light noises that might cause erroneous readings from the sensor.

- Photodiodes

The photodiodes are connected to pin 2 and 3, which are set to analogue input pins in the program. It is connected across the 5V rail to ground with a large resistor in series to provide the signal for the ATtiny.

A photodiode's output is its reverse saturation current. It is thus wired in the reverse direction with the cathode connected to 5V and anode towards ground. When it detects a high concentration of photons, the photoelectric effect triggers, sending electrons across the diode in reverse direction. This means light intensity is proportional to the current through the device. This is however a very small current, in the order of tens of microamperes. For this signal to be readable by the microcontroller, the current is passed through a 100kΩ resistor. By Ohm's law, this means a voltage of around 3V will be generated and can thus be sent to the ATtiny.

The diode is connected from the 5V rail because that provides a constant voltage, unlike the 6V rail which has a voltage that may decrease as the batteries get depleted.

4.3.2 Circuit – Output stage

This stage consist of a buzzer, and two motors. They are all in one way or another connected from the output pins of the ATtiny microcontroller.

- Buzzer

Buzzer is consider to be the Superbug function of our device so it will be explained in details in section 5.1.

- Motors

The motors are connected to pin 5 and 6, which are set as analogue output pins in the program. The microcontroller is unable to supply a current sufficient for the motors, thus it is first amplified with a transistor.

It is also connected in parallel with a capacitor. This is a by-pass capacitor, used to clean up the AC noise around the motor allowing only a pure DC voltage to pass through the other components. This means that the use of the capacitor is to protect other components from the AC noises (spikes) produced by the motor.

A common emitter setup is used in the circuit. This allows the voltage output from the ATtiny to be directly translated into the current through the motor.

The voltage output is controlled by a Pulse Width Modulation (PWM) signal at 5V from the microcontroller. This is used to mimic an analogue output, however instead of varying the DC voltage, it generates a square wave of varying duty cycle. An experiment is carried out to determine how the duty cycle may affect the speed of the motor. The detail of this experiment is described under section 5, testing of the bug.

4.3.3 Circuit – Miscellaneous

For the smooth running of the circuit, additional components are installed to support the main functionality of the Bug.

- Decoupling Capacitors

Capacitors of 0.1 μF are installed across all power rails to protect the circuit, especially the microcontroller.

All the wiring on the breadboard has unwanted resistance and inductance. This becomes significant when a PWM output is switching between high and low at high frequency. It can be difficult for the ATtiny or the voltage regulator to handle. Thus to prevent the microcontroller from malfunctioning or accidentally resetting, the decoupling capacitors are introduced to smooth out the voltage signals.

- Resistor across VCC and RESET pin

The 10kΩ resistor from VCC of ATtiny to its RESET pin is used to pull up its voltage, preventing the microcontroller from resetting spuriously.

- Voltage Regulator

A 5V voltage regulator is installed between the 6V rail and ground, providing a 5V rail in the circuit. This is necessary because many of the components in the circuit would benefit from this, especially the ATtiny.

From measurement this voltage can reach up to 6.6V for 4 AA batteries in series. However the microcontroller is rated only up to 5.5V, thus for safety its VCC pin is connected to the 5V rail.

The photodiodes benefit from this arrangement too. The voltage supplied from the batteries decreases as they get depleted. Since the voltage regulator provides a stable voltage that is unaffected by the draining battery, the photodiodes can provide a consistent signal input to the microcontroller.

- Fuses

Fuse is used to protect the other components of the circuit from high currents. It is a ceramic body that consists of a metal wire which has low breaking capacity. When it gets hot, which means that high current pass through it, it breaks and this results an open circuit, protecting in that way the other components.

4.4 Discussion of Code

The code is written as an Arduino sketch and uploaded to the ATtiny with Arduino Uno as the ISP. This allows our group members to code in a language with more familiarity.

Stages of the code

The program can be broken down into a few stages.

4.4.1 Code – Setup

In the setup stage, different pins are given their functions. By the anticlockwise convention, pin 1 stays as the reset pin, pin 2 and 3 are set as analogue inputs. Pin 5 and 6 are analogue outputs, while pin 7 is a configured as a digital output pin. Pin 4 and 8 are not specified because they are fixed with 8 being V CC and 4 being the ground pin. The pin numbers may differ from the ones written in the code because as it has been mentioned, the program will first run on an Arduino, which has a different pin layout.

Certain variables are defined in this stage as well. This is to make adjusting the code easier between experiments. The speed of motor is defined to have three levels, high, normal, and low. This corresponds to different duty cycles, and they are used to control the movement of the bug.

The values of the voltages obtained from the photodiodes can fluctuate within a limited range, which will be considered noise. Noise may affect the functioning of the Bug. This is why the values of the voltages used in the program differ by an average of 0.4 V from the voltages obtained at each position. Now the program will be sure that the Bug will be on the position that the voltages say it is. For example, when the left sensor detects a line a voltage of 2.4 V appears. In this case the program will use a value of 2.8V just in case any noise affects the reading. 2.8 Volts is still very bellow the voltage read when the Bug is totally on a white surface (4.12V). The final result with this approach is a Bug that runs smoother and with higher consistency.

4.4.2 Code – Line Following

The bug begins its function by first following the black line. This is accomplished by having the ATtiny constantly adjusting the speed of the motor depending on what is detected from the two photodiodes.

A loop is constructed to achieve this function. Within each iteration, the microcontroller will first get a reading from the photodiodes using the analogRead() function. This returns an integer between 0 and 1023, which corresponds to a voltage between 0 to 5V. The accuracy of this can be calculated as 5/1024 = 4.9mV/unit. This is more than sufficient for a line

follower, and thus bands were introduced in the setup stage. The integer is then manipulated through an algorithm that maps the input bands to corresponding outputs to the motors. This output value is pushed to the analogue output pins with analogWrite() function. The function provides a PWM wave at approximately 490 Hz, with varying duty cycles. From empirical research, we have discovered that the motors are the most controllable at around 35% duty cycle. This corresponds to an analogWrite() value of $0.35 \times 255 \approx 90$. The details of the experiments carried out are specified in the later sections of the report.

To illustrate with an example, suppose the bug has deviated slightly to the right of the line. This means the left diode would detect less brightness than its counterpart on the right. Assuming there it is 70% black on the left and 20% on the right, this maps to around 0.3V across the left resistor and 1.1V right. A new iteration begins at this exact moment, and analogRead() function obtains $0.3 \times (1024/5) \approx 60$ on the left and similarly $1.1 \times (1024/5) \approx 225$ on the right. This is mapped through the algorithm, which sets the right output to high, while the left output to low, allowing the bug to turn left. Performing exactly what is required to maintain the bug's position on the line.

4.4.3 Code – Go Straight

Once the bug exits the line, both of its sensors would detect white, or less than 10% blackness. As soon as this reading is maintained after a few iterations, the bug breaks out from the loop and starts its next series of tasks. This stage is relatively simple, and consists only of setting both motors at normal speed, and a delay() function that pauses the program for a certain period. The precise value of this time is determined empirically, by letting the bug run and measuring how long it would take to travel straight for 10 cm. It is important to check if it is going through a black border so that it stops. In order to achieve this the delay function is divided into 10 parts. After each delay is done the program will check if the sensors are detecting a black surface.

4.4.4 Code – Spiral

To perform the spiral after going straight for 10 cm, the microcontroller sets the speed of the left motor to low speed, while gradually increasing the speed on the right motor. This results in an anticlockwise spiral movement of increasing radius. There are two kinds of spirals, those which have a constantly increasing radius and those that have an increased radius every time a half circle path is done. The designed code follows the second approach.

4.4.5 Code – Miscellaneous

To prevent unwanted effects from happening, a few additional blocks of code are added into the program.

In the unlikely case that the bug reached the border of the playing field, the sensors would detect a black surface. If this happened, the bug would stop immediately to prevent itself from exiting the designated area.

4.4.6 Code – Buzzer

Since it is a Superbug function, it will be explained in section 5.1.

4.5 Breadboard Layout

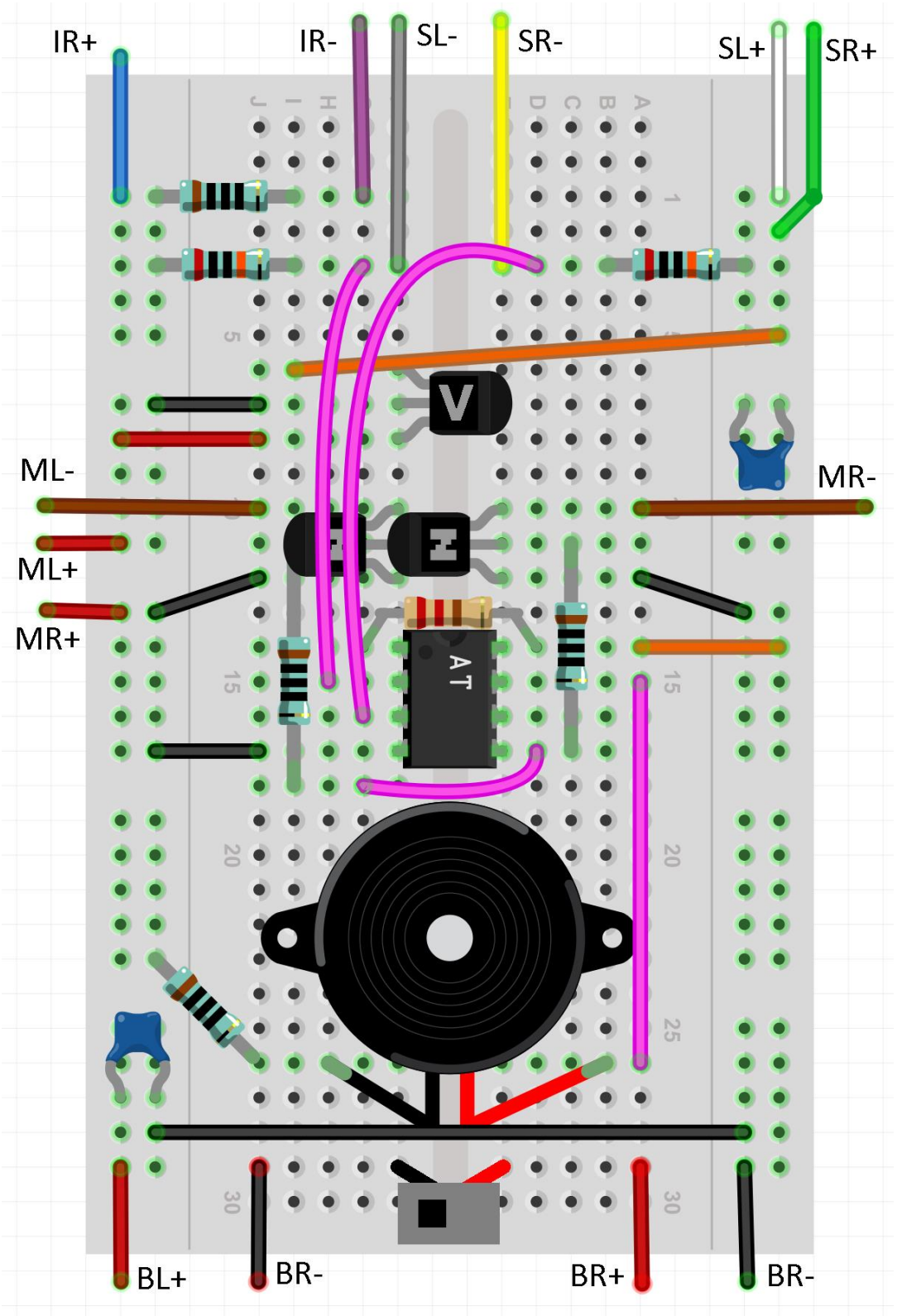

Figure 9 Breadboard Layout

4.5.1 Layout of breadboard

This layout is not fully representative of the actual design due to software limitations. The rough size and orientation of components are mostly accurate. Thus it does provide a good approximation for how the circuit diagram is realised onto the breadboard. The most significant difference between this illustration and the actual bug is of the buzzer. On the actual bug, the buzzer used is installed on to the breadboard with through-hole pins, instead of a pair of red and black wire as shown in the diagram.

Label	Meaning	Connects to
IR+	Infrared LED	Anode of LED
IR-		Cathode of LED
SL+	Left sensor	Cathode of photodiode
SL-		Anode of photodiode
SR+	Right sensor	Cathode of photodiode
SR-		Anode of photodiode
ML+	Left motor	
ML-		
MR+	Right motor	
MR-		
BL+	Left battery	
BL-		
BR+	Right Battery	
BR-		

Left most rail is used as 6V supply, while the right most rail is used as 5V supply. The two rails near middle are used as ground rails.

For the NPN transistors, brown wires connects from the collector pin to the motors, and black wires connects the emitters to ground. The base pin is the middle is send to the microcontroller through a pair of 100Ω resistors. V in the diagram stands for a voltage regulator, which has input connected to 6V from a red wire and its output setting the right rail at 5V through an orange wire.

4.6 Engineering Insights

4.6.1 Choosing the size of resistors

Choosing the size of current limiting resistors

The LEDs can only take a certain level of current before breaking down. To prevent this from happening, a current limiting resistor is necessary. From the LED data sheet, the rated

maximum current is of 100mA, with its forward voltage drop of 1.6V, it is possible to calculate the required size of resistor.[2]

$$R_{lim} = \frac{V_{CC} - V_{LED}}{I}$$

Suppose we want around half of the rated current, by simple substitution R is found to be around 100Ω.

Resistor for buzzer

The size of the resistor in series with the buzzer is determined empirically by observing the waveform of the voltage drop across the resistor while varying its magnitude. By the Ohm's law, a lower resistor implies a lower consumption of power. Thus it is chosen as the lowest possible resistor value that is still sufficient to limit the abrupt changes in voltage to a reasonable value that will not damage the other components.

4.6.2 Failures

A pair of infrared receivers were first purchased as the sensors for the bug. Each receiver has a demodulator in its internal circuit. This means it decodes signals with a carrier frequency of 38 kHz, instead of simply detecting the brightness of the infrared light it sees. Thus the receivers were discarded, and replaced with a pair of infrared photodiodes. This was caused by a series of misunderstandings of the component's nature, but it proved to be a valuable learning experience.

Due to the delicacy of the components, these photodiodes were broken during the experiments. To determine the best position for the diodes to be at, the legs were bent excessively, which resulted in the snapping of the legs. In order to fix this, stripped wires were soldered onto the broken pins. However, this had the side effect of overheating the semiconductor, causing the diodes to have unexpected behaviours in the following experiments. Thus a new pair of diodes were purchased to replace the broken ones.

5 EEBug enhancements and Testing

5.1 Extra functionality – Buzzer

The buzzer is simply connected to pin 7, set as a digital output pin in the program, and in series of a current limiting resistor for safety reasons. By varying the frequency of switching the pin voltage high and low, a square wave is provided to the buzzer, thus allowing a tune to be played. While the hardware allows any note from 31 Hz to 65535 Hz, the buzzer is rated for highest efficiency around 2000 Hz. This means given a square wave of varying frequency but constant amplitude, the sound produced would be the loudest when the frequency is around 2000Hz.

Any frequency of a note in twelve-tone equal temperament can be calculated with this equation,

$$f_n = f_a\left(\sqrt[12]{2}\right)^{n-a}$$

Where f_a is the frequency of the reference note a in Hz. By convention, this is chosen to be A_4 at 440Hz. n-a is the number of notes n is above a.

Thus to maximise the efficiency of our bug, we have created our tune around the seventh octave (A_5~A_7). These notes have frequencies around the resonance frequency of the buzzer (880~3520 Hz).

Numerous debates have been carried out in the team in deciding what song to be coded in the microcontroller. Due to the texture of the instrument and the small memory space available in the chip, we have agreed upon that choosing a jingle that is short and sweet should be the best option.

5.1.1 Code

The buzzer is set to play a short tune as it starts off, and another when it has accomplished all its tasks. In coding terms, this is done by the function tone().[3] The function generates a square wave at any specified frequency between 31 to 65535 Hz. To create a song from this, the music is first converted from manuscript form to a list of frequencies and durations. The conversion is done through the aforementioned formula in the buzzer section of this report. This list is then written into ATtiny so that the tune can be played.

5.2 Measurements and expectations

5.2.1 Change of voltage in the sensors with brightness

The code that makes the Bug work uses measured values of voltages so that it knows where the line is. As it can be seen in Figure 8 the brighter the surface, the more voltage there is. To make these measurements the Bug was put on a sheet of paper with scaled brightness from white to black. At the beginning there is a slight mismatch between both sensors. However, it disappears as the surface gets darker.

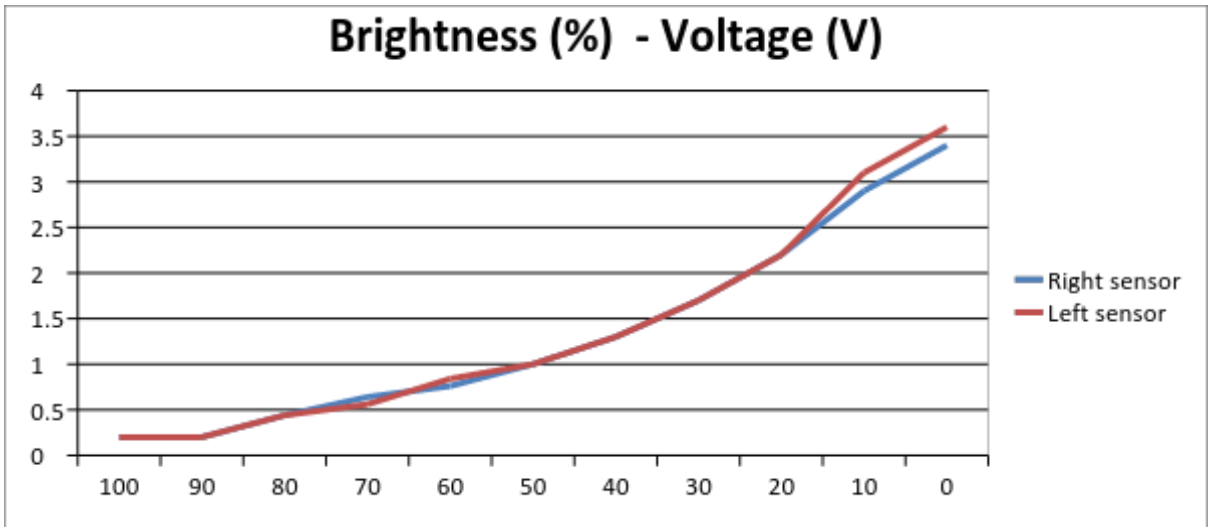

Figure 10 Voltage-Brightness

5.2.2 Speed of the Bug as a function of the duty cycle

As mentioned earlier, the output of the microprocessor uses PWM. It is because of this that a measurement of the speed of the motor with different duty cycles was necessary. As it can be seen in figure 9, speed is proportional to duty cycle. In order to keep the bug within a reasonable speed, a duty cycle around 30% is used.

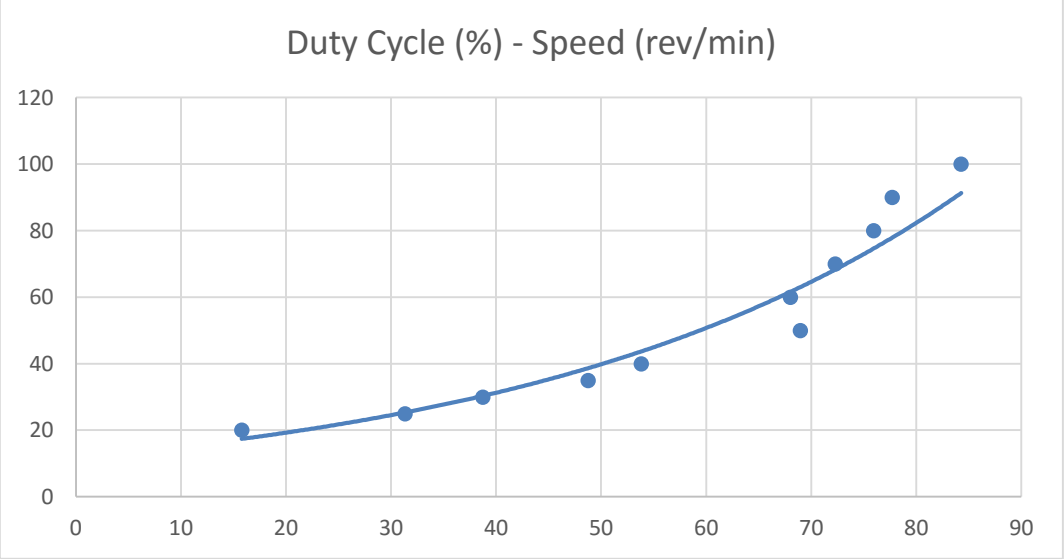

Figure 11 Duty cycle-Speed

5.2.3 Voltage as a function of position

The values of the voltages have already been measured with changes in brightness. However, it is still necessary to check if they are still the same when detecting a line. It has to be taken into account that there will be a certain mismatch between both measurements due to a different angle of incidence of the light in the room. Also these measurements were taken using the voltage supplied by the batteries themselves instead of by an external supplier. The voltage was 5.9 volts.

As it can be seen by the following table the voltage when the Bug is on the line is just halfway between the voltages on black and white surfaces. This is useful information as it will be the numbers to be plugged into the code that runs the device.

It can also be perceived that there is a slight mismatch in the voltages of both sensors. This is due to the fact that the sensors are relatively big and in order to fit they have to be bent. This leads to a slight difference in the angle between both sensors and the ground and therefore they do not perceive the same amount of IR.

	On line	On white	On Black
Right sensor	1.898	3.78	0.416
Left sensor	2.434	4.12	0.668

5.3 Troubleshooting

The Bug is an electronic device and as such it is easy to make a wrong connection. This could lead to malfunctioning of the Bug. In the following lines common sources of problems will be explained and how to solve them.

5.3.1 Fuses

If one or both of the motors stop moving it could be due to a short circuit that made excessive of current go to the motor. In order not to brake it fuses have been introduced into the device. If this is the case there will no longer be a thin wire in the fuse. To solve this problem check the connections in the breadboard again and then replace the fuse.

5.3.2 Correct position

The Bug is an accurate device and as such any slight change in the amount of radiation it perceives could lead to malfunctioning. In order to avoid this, ensure that the Bug is allocated just in the middle of the line.

5.3.3 Adjust angle

The emitter and sensors are connected to the Bug by very thin wires. These are easily bended which allows us to change the angle of the sensor with respect to the ground. If one or both

of them look too far away from 90° adjust them carefully as they may break. If this does happen order a new component (the same one) and introduce it through two holes in the white box of the Bug. The length of these pins may be adjusted appropriately by cutting them.

5.3.4 Batteries

Batteries run out very easily, check how charged they are. They should always be above 5.4V. Some batteries such as Duracell tell how much battery is remaining just by looking at their side. Otherwise a DMM is required. Connect two cables to each of the sides of the battery and check that the voltage shown on the screen is above 5.4V. In case of doubt or if none of these elements are available replace them by a new set.

5.3.5 Correct connections

It may happen that during use one of the connections becomes loose, a component had to be replaced or by any reason the user decides to move the components around. In order to avoid malfunctioning take a look at the previous figure showing the layout of the breadboard and check that all the components and cables are in the right place.

5.3.6 Switch

Remember to turn the switch on. Move the slide and wait for a few seconds as the device may take some time to initialize. Do not play with the switch as it is a mechanical structure and therefore it may break easily. If this happens order the same switch and make the appropriate connections to the breadboard. The user may need to attach a cable to the switch.

6 References and Data Sheets

6.1 References

[1] Sparkfun. (2016) *Tiny AVR Programmer Hookup Guide.* [Online]. Avaliable from: https://learn.sparkfun.com/tutorials/tiny-avr-programmer-hookup-guide/attiny85-use-hints

[2] Electronics Club (2016). *LEDs.* [Online]. Avaliable from: http://electronicsclub.info/leds.htm

[3] Arduino.cc (2016) *Play Melody.* [Online]. Avaliable from: https://www.arduino.cc/en/Tutorial/PlayMelody

6.2 Data Sheets

Infrared Receiver (not used in final design)
http://www.farnell.com/datasheets/1699654.pdf
Infrared Emitter
http://www.farnell.com/datasheets/1866444.pdf
Microcontroller
http://www.farnell.com/datasheets/1693899.pdf
Infrared Photodiode
http://www.farnell.com/datasheets/1672046.pdf
Voltage Regulator
http://www.farnell.com/datasheets/1642410.pdf
Piezo Sounder
http://www.farnell.com/datasheets/1851881.pdf
Slide Switch
http://www.farnell.com/datasheets/1839844.pdf

Appendix I – Arduino Code

```
  int LeftMotorPin = 11;            //PWM output
  int RightMotorPin = 10;           //PWM output
  int LeftSensorPin = 3;            //Analog input
  int RightSensorPin = 2;           //Analog input
  int BuzzerPin = 7;                //Buzzer output
  int AnalogLeft = 0;               //Value of read voltage at the left sensor
  int AnalogRight = 0;              //Value of read voltage at the right sensor
  int HighSpeed = 76;               //High duty cycle
  int SlowSpeed = 51;               //Low duty cycle
  int NormalSpeed = 63;             //Medium duty cycle
  int TimeFor10cm = 3000;           //Time it takes in ms to make 10 cm with HighSpeed on
both motors
  int count;                        //Used so that in void loop it always reads the voltages
again before doing something different
  int spiral = 0;                   //Used to tell whether or not do the spiral
  int timer = 0;                    //Used to divide the TimeFor10cm into parts and check
if there is a black surface (therefore stop)
  int songLength = 18;              //Length of song
  char notes[] = "cdfda ag cdfdg gf "; //space represents rests
  int beats[] = {1,1,1,1,1,1,4,4,2,1,1,1,1,1,1,4,4,2};    //rhythm of the song
  int tempo = 113;                  //Tempo of the song

void setup() {
  // put your setup code here, to run once:
  pinMode (LeftMotorPin, OUTPUT);          //Left motor pin declared as output
  pinMode (RightMotorPin, OUTPUT);         //Right motor pin declared as output
  pinMode (LeftSensorPin, INPUT);          //Left sensor pin declared as input
  pinMode (RightMotorPin, INPUT);          //Right sensor pin declared as input
  pinMode (buzzerPin, OUTPUT);             //buzzer pin declared as output
}

void loop() {                              //this code will run infinitely many times
  count = 0;                               //Reinitialize count
  AnalogLeft = analogRead(LeftSensorPin);  //read the voltage at the left
  delay(1);                                //delay for stability in the reading
  AnalogRight = analogRead(RightSensorPin);
  delay(1);                                //delay for stability in the reading
  if (AnalogRight < 491 && AnalogRight > 205 && AnalogLeft < 573 && AnalogLeft > 246 &&
spiral == 0){                              //It is on the line
    analogWrite(LeftMotorPin, NormalSpeed); //Run at normal speed
    analogWrite(RightMotorPin, NormalSpeed);//Run at normal speed
    count++;                               //Do not do anything else in this loop (except
checking if there is a black surface)
    delay(200);                            //wait for 200 ms
  }
  if (AnalogLeft > 573 && AnalogRight < 491 && count == 0 && spiral == 0){
//Left sensor sees white
    analogWrite(LeftMotorPin, NormalSpeed); //Normal speed at the left
    analogWrite(RightMotorPin, SlowSpeed);  //Slow speed at the right
    count++;                               //Do not do anything else in this loop (except
checking if there is a black surface)
    delay(200);                            //wait for 200 ms
  }
  if (AnalogRight > 491 && AnalogLeft < 573 && count == 0 && spiral == 0){
//Right sensor sees white
```

```
    analogWrite(RightMotorPin, NormalSpeed);//Normal speed at the left
    analogWrite(LeftMotorPin, SlowSpeed);   //Slow speed at the right
    count++;                                //Do not do anything else in this loop (except
checking if there is a black surface)
    delay(200);                             //wait for 200 ms
  }
  if (AnalogLeft > 675 && AnalogRight > 634 && count == 0 && spiral == 0){
//White surface
    analogWrite(LeftMotorPin, HighSpeed);   //High speed at the left
    analogWrite(RightMotorPin, HighSpeed);  //High speed at the right
    count++;                                //Do not do anything else in this loop (except
checking if there is a black surface)
    delay(TimeFor10cm / 10);                //Divide the time for 10 cm into 10 so that in
each division it checks if there is a black border
    timer++;                                //Tells us when to stop delaying
  }
  if (timer > 9){                           //if it goes above nine, 10cm will have been done
and the spiral will begin
    spiral++;                               //make spiral = 1
    timer = 0;                              //reset timer so that it does not do this "if"
statement again
  }
  if (spiral > 0 && spiral < 26){           //do it 25 times
    analogWrite(LeftMotorPin, SlowSpeed);   //write slow speed
    analogWrite(RightMotorPin, 64);         //increment speed at right motor
    delay(200);       //200*25 = 5 seconds. Always check if there is a black border
    spiral++;         //increment the spiral counter (greater than 25 exits the function)

  if (spiral > 25 && spiral < 51){          //do it 25 times
    analogWrite(LeftMotorPin, SlowSpeed);   //write slow speed
    analogWrite(RightMotorPin, 77);         //increment speed at right motor
    delay(200);       //200*25 = 5 seconds. Always check if there is a black border
    spiral++;         //increment the spiral counter (greater than 50 exits the function)
  }
  if (spiral > 50 && spiral < 76){          //do it 25 times
    analogWrite(LeftMotorPin, SlowSpeed);   //write slow speed
    analogWrite(RightMotorPin, 90);         //increment speed at right motor
    delay(200);       //200*25 = 5 seconds. Always check if there is a black border
    spiral++;         //increment the spiral counter (greater than 75 exits the function)
  }
  if (spiral > 75 && spiral < 101){         //do it 25 times
    analogWrite(LeftMotorPin, SlowSpeed);   //write slow speed
    analogWrite(RightMotorPin, 105);        //increment speed at right motor
    delay(200);       //200*25 = 5 seconds. Always check if there is a black border
    spiral++;         //increment the spiral counter (greater than 100 exits the function)
  }
  if (spiral > 100){                                //the spiral finishes
    analogWrite(LeftMotorPin, 0);                   //switch off left motor
    analogWrite(RightMotorPin, 0);                  //switch of right motor
    int i, duration;
    for (i = 0; i < songLength; i++){               //step through the song arrays
      duration = beats[i] * tempo;                  //length of note/rest in ms
      if (notes[i] == ' '){                         //is this a rest?
      delay(duration);                              //then pause for a moment
      }else{                                        //otherwise, play the note
      tone(buzzerPin, frequency(notes[i]), duration);
      delay(duration);                              //wait for tone to finish
```

```
    }
    delay(tempo/10);                              //brief pause between notes
  }
//This function takes a note character (a-g), and returns the
//corresponding frequency in Hz for the tone() function.
  int frequency(char note){
  int i;
  const int numNotes = 8;
  char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
  int frequencies[] = {262, 294, 330, 349, 392, 440, 494, 523};
  for (i = 0; i < numNotes; i++){
    if (names[i] == note){
      return(frequencies[i]);
    }
  }
  return(0);
}
  if (AnalogRight < 205 && AnalogLeft < 245){      //Black  surface.  This  is  going  to  be
checked every time the loop is executed
    analogWrite(LeftMotorPin, 0);                 //switch off left motor
    analogWrite(RightMotorPin, 0);                //switch off right motor
    tone (buzzerPin, 2000, 2000);                 //plays error tone
  }
}
```

Appendix II – Low level explanation of the code

At the beginning of the program all the variables are initialized and the pins are set as inputs or outputs. HighSpeed, SlowSpeed and NormalSpeed are set to 76, 51 and 63 (remember how the duty cycle works in arduino). TimeFor10cm has been set to 3 seconds.

Void loop starts and with it count is reset. AnalogLeft and AnalogRight correspond to the voltage readings at sensors left and right respectively. It takes some time in the order of microseconds for the processor to read the voltages at the inputs. It is because of this that a delay of a millisecond is used as it gives a higher reading stability.

The voltages set to be the margin at which there is a line or a white surface were measured and then converted to arduino values. As already mentioned, the final numbers plugged into the program differ from these in order to avoid any noise.

The first if statement checks if the Bug is on the line by comparing the read voltage and the expected voltage. Also, it will not do this algorithm if spiral is bigger than zero. The value spiral will be non zero when the Bug is doing a spiral. If it is the case that the Bug is on a line it will start the motors with a normal speed. A delay of 200 ms is applied so that it runs at this speed and direction for some time. The integer count is added one in order not to do any of the other functions, except checking if there is a black border line, which would stop the Bug immediately. After all this processing the loop is restarted and the integer count is reset.

The next case studied by the program is if the left sensor sees white. This means that the Bug is slightly deviated from the line to the left. As always voltages are compared and in order for this function to be performed the integer count has to be zero, which means that the Bug is not on the line (if not count would have been incremented by one).

The same procedure is done for the right sensor seeing white and for both sensors seeing this brightness. However, for the last case, if both sensors see white it means that there is no more line and therefore ten centimeters have to be done. There is a slight problem, what if a black border is reached before this happens? In order for the processor to know it, the delay is splitted into ten other delays. Each time the it finishes, the ATtiny will check that there is no black border.

After the 10 delays finish, the spiral will begin and the integer spiral will be equal to one in order to let the following functions work and stop the ones used before.

The spiral algorithms are divided into four parts. In each part the left motor has SlowSpeed and the right's speed is incremented. A delay of 200 ms is applied. As before, every time the function is done, the program will check that it has not reached a black border. With each change in the speed of the right motor the radius of the path is increased significantly.

Finally, after the spiral has been completed the motors will be switched off and an infinite while loop will appear that will run a useless code based on addition and subtraction of the integer count.

This last function is relatively similar to what would happen if the black border is detected. This code is very important as it is used all the time during the execution of the program. It will compare the voltages as before and in case the conditions are true, the motors will be switched off and an infinite useless loop will be executed.

Appendix III – Components Ordered

Supplier	Component ID	Component Official Name	Quantity	Price per component	Total Price	Order Date
Onecall	1455162	ATTINY85-20PU 8 Bit Microcontroller	1	1.05	1.05	12/02/2016
Onecall	1612496	VISHAY TSAL6200... Infrared Emitter, High Power	2	0.36	0.72	12/02/2016
Onecall	4913139	VISHAY TSOP34838 Infrared Receiver, Remote Control	2	0.52	1.04	12/02/2016
EEDStores	SD0190	TRANSISTORS BJT 2N3904	2	0.15	0.3	12/02/2016
Onecall	9489460	LP2950ACZ-5.0 Fixed LDO Voltage Regulator	1	0.47	0.47	18/02/2016
Onecall	1212743	SFH203-FA PHOTODIODE	2	0.48	0.96	18/02/2016
Onecall	2443201	MURATA PKM22EPPH2001-B0	1	0.72	0.72	01/03/2016
Onecall	1212743	OSRAM SFH203-FA PHOTODIODE	2	0.49	0.98	01/03/2016
Onecall	2435097	C & K COMPONENTS OS202013MT5QN1 Slide Switch	1	0.35	0.35	18/03/2016