B37VB - Edinburgh - Robotics - Group 3 - Final Report

Torin Stanton-Andersson (H00438492) and David Mariani (H00438981)
ts2104@hw.ac.uk - dm2166@hw.ac.uk

14/04/2024

Table of Contents

Revision History

Date	Author Name	Version Number	Notes
14/04/2024	David Mariani	1.0	
14/04/2024	Torin Stanton-Andersson	1.0	

1. Introduction

This report details group 3's Robotics project for the course B37VB. The final aim of this project is for group 3 to edit and program a buggy to enable fully functional light tracking as well as any additional features they deem appropriate. For this project the Arduino UNO board was used with the ATMEGA328P microchip.

2. Theory of Operation

The Arduino is capable of creating an output of 5 volts (drawing power from the onboard battery) which was used to operate the motors and components. The buggy used an Arduino UNO mounted on a platform with a breadboard prototyping area in front for any components that would need to be added. There was also a USB connection port/Power connector and connections for the H-bridge. The motors, their wheels and a power source were also mounted on the underside of the platform and connected to the Arduino via a USB connection port. The H-bridge (DRV8833 Dual H-Bridge Motor Driver) connections were located above the motors (on top of the board) for easy access. The breadboard's power and ground lines were connected to the Arduino's 5V power port and ground port respectively. Components would be fed with a power wire on one side with a voltage divider setup and a wire connected to ground on the other in order for current to pass. By uploading code to the Arduino, the flow of current through any motors and components connected to it could be controlled through a number of pins on the UNO.

PWM Motor Control

Pulse width modulation is a current control technique that allows for efficient variable control of motors. It works by pulsing DC current with different lengths of time depending on how much current is required. This creates a number of square waves with different widths (more current is given by longer widths). This is done by the motor controller using a low pass filter to create an average of the PWM value. In order to check the PWM values, one motor at a time was hooked up to an oscilloscope and ran on all of its different speed/power settings while measuring its output. Low values created pulses of a smaller width, medium values nearly doubled the width while the maximum value would just be a continuous pulse with no breaks.

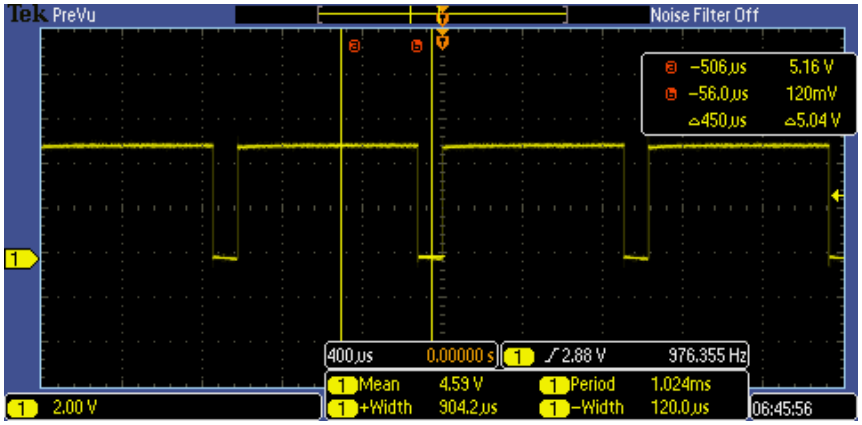

Figure 1: Left engine mid value.

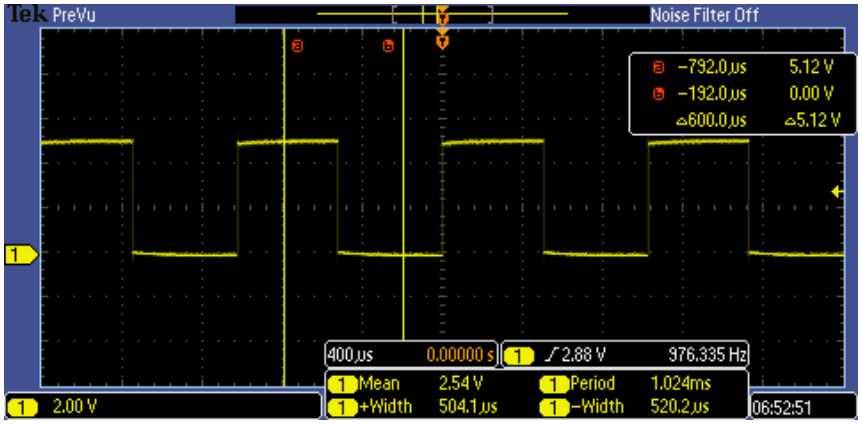

Figure 2: Left engine low value.

Open Loop Motor Characterisation

Initially, in order to ensure the consistency and functionality of any future additions to the buggy, the operation of the DC motors had to be characterised. This involved using PWM signals to control the motor speeds with a max PWM value of 255 for maximum speed. In order to reverse, a negative value would have to be used but since a negative signal isn't possible the code would simply switch the values going into the H-bridge control pins causing the wheel to rotate in the opposite direction. At first, it was discovered that the motors were not running at the same speed. When both were set to the same value the buggy would turn left, this indicated that the left motor was weaker than the right one. In order to find the proper speed to run the left motor at, the code was edited to maintain the right motor at the same speed while slowly ramping the speed of the left one up after certain intervals. A flashing light would indicate moments where the speed changed, using this method one could wait until the robot began driving straight and then record the value for the corresponding left PWM value. This process was repeated to find the corresponding left motor PWM values for small, medium, and large values of the right motor. A speed test was performed and the time taken for the buggy to travel 1 metre was averaged, which gave the buggies average speed at each of the different speed settings.

LDR Circuit

The Light dependant resistor (LDR) circuit was simply assembled using 2 LDRs, 2 resistors and a number of wires connecting the components to their allocated pins or ground.

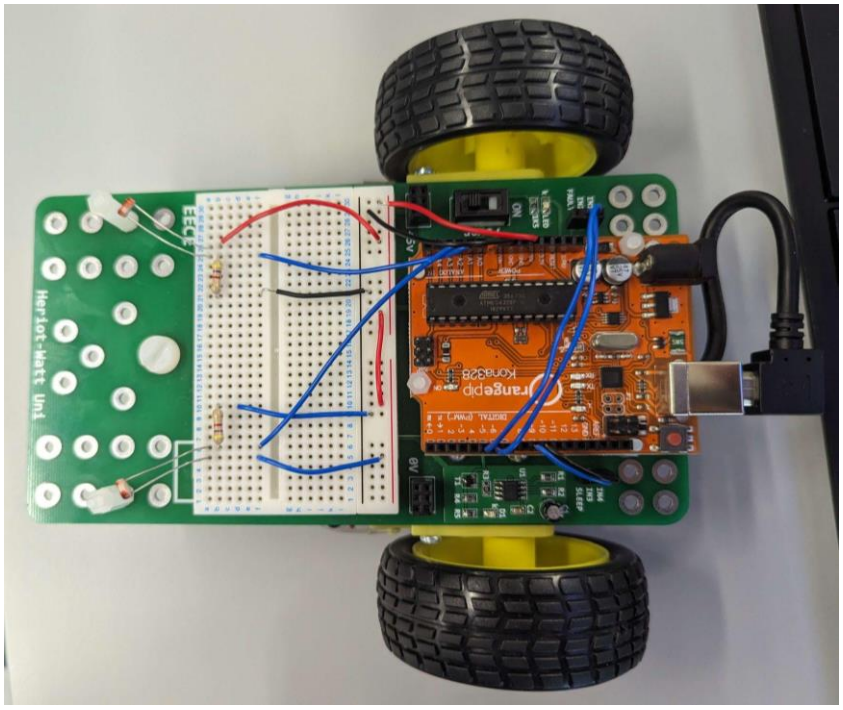

Figure 3: Simple two LDR circuit on the buggy.

Resistors had to be carefully chosen in order to optimise the function of any Light emitting diodes (LED's) or LDRs used. Not using a resistor as a voltage divider could potentially cause a component to "Burn out" due to the excess current flowing through the component. Multiple resistor values were used in order to find which would best serve the components. Resistors were also used to alter the range of the values output by the LDRs.

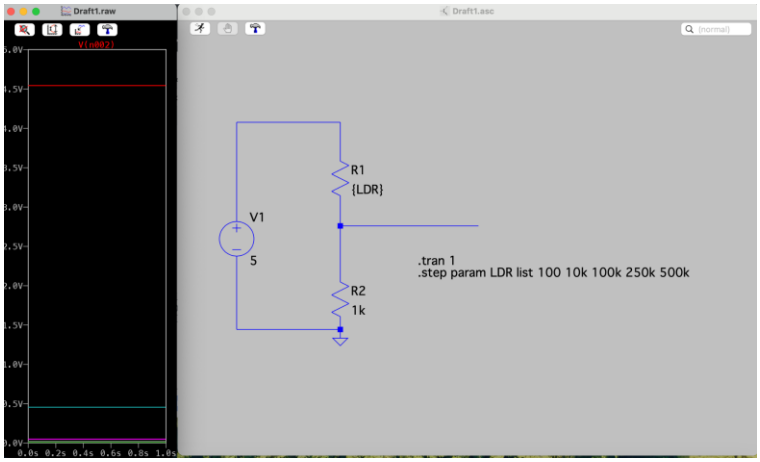

Figure 4: Simulation of the LDR-ADC circuit using a 1k Ω resistor.

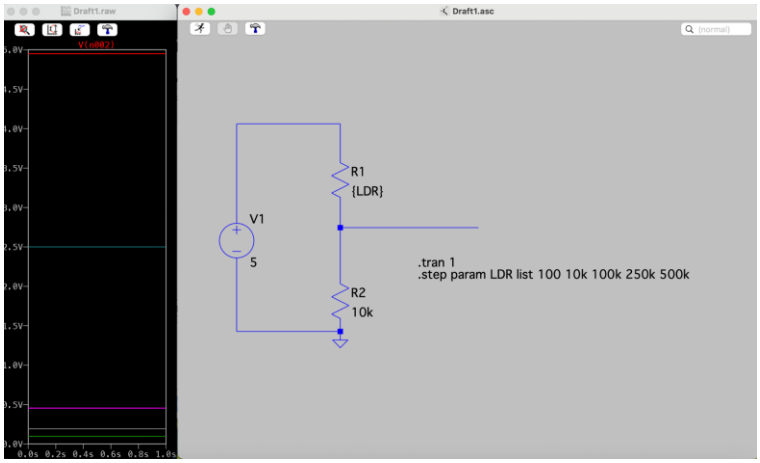

Figure 5: Simulation of the LDR-ADC circuit using a 10k Ω resistor.

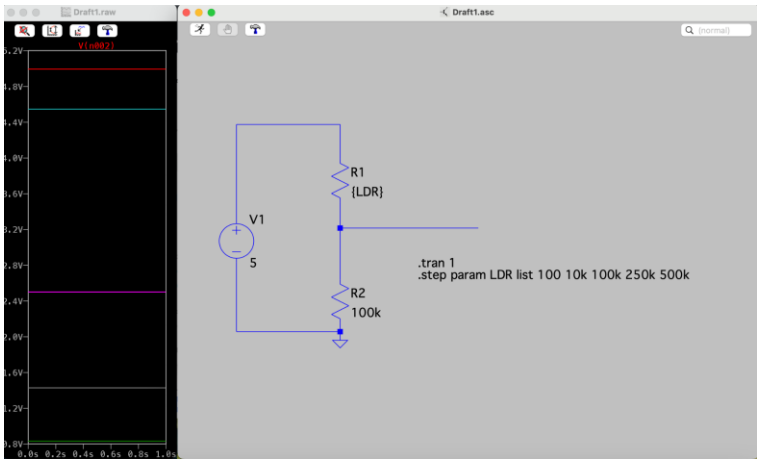

Figure 6: Simulation of the LDR-ADC circuit using a 100k Ω resistor.

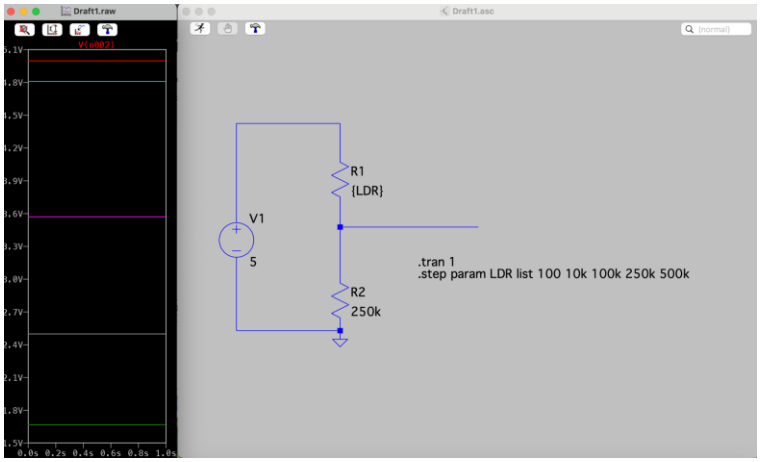

Figure 7: Simulation of the LDR-ADC circuit using a 250k Ω resistor.

The UNO uses a 10-bit ADC (Analogue to digital converter) which allows for an output range of 1024 bits (from 0 to 1023) and a full-scale voltage of 5V. Using the LDRs It would output a range of values from 100 - 300 for low light intensities, 300 - 600 for medium light intensities and 600 - 1023 for High light intensities. The best resistor to use is the one with the largest range of ADC values, Table 1 indicates that either the 1k or 4.7k resistor is the best.

Table 1: Values of the resistance of different resistors and their corresponding output ADC value when dark and light/bright.

Resistor Value (Ω)	ADC output - Dark	ADC output - Bright
100	120	650
1k	140	950
4.7k	160	1010
10k	280	1020
100k	760	1023
220k	900	1023

3. Light-Tracking Algorithm

Once the buggy is turned on, using the power switch, it will begin driving in a straight line until it finds a light source to follow. The light tracking algorithm utilised 2 LDRs that would constantly measure the intensity of incident light, if the intensity on either LDR exceeds a certain check threshold (which is made necessary due to the presence of ambient light) the program will calculate which LDR is subject to a higher intensity and then change the power values of each wheel to preset values in order to execute a turn - this would only occur if the difference in ADC output values between the two LDRs exceeded 150. The wheel opposite to the direction of light will speed up and the other wheel will slow down causing the buggy to turn towards the light. Once the values equalised or fell below the threshold value the buggy would go back to driving straight. If the ADC output value of either LDRs rose above the maximum threshold of 750, the wheels would stop until the value dropped below the threshold and then resume operation. This was done in order to prevent the buggy from colliding with the light source if the source stopped moving.

Table 2: Motor turn PWM value presets.

Motor side	Forward PWM	Right turn PWM	Left turn PWM	Hard left turn PWM	Hard right turn PWM	Reverse PWM
Right	100	75	175	150	-75	-150
Left	127	222	95	-95	190	-190

4. Hard Turn Feature

When the base light tracking implementation was functionalized, the hard turn feature was added. This involved the addition of 2 more LDRs facing towards the left and right of the buggy and modifying the light tracking code to account for the extra inputs. The addition of these LDRs allowed the buggy to differentiate the angle of light source on either side and the conditionals used to differentiate the direction of the light source had to be made more complex. The buggy would follow a light source as it did previously but if the intensity of light incident on the side LDRs exceeded the front LDRs by a given threshold of 150, the buggy would perform a hard directional turn - This was done by causing the wheel opposite to the source to speed up while the wheel closer to the source would slowly reverse.

Table 3: Motor hard turn PWM value presets

Motor side	Hard left turn PWM	Hard right turn PWM
Right	150	-75
Left	-95	190

5. Conga line

Once work on the light tracking implementation was finished, work on the conga line algorithm was able to begin. This involved modifying the buggy with a light source facing backwards (A 12V 540 mA LED was used) as well as a number of code additions to the function of the LDRs. This particular implementation required a blinking light source; in order to do this a BC337 NPN transistor component was set up on the breadboard and connected to the bulb. The transistor enabled the UNO to control the flashing of the light source. The BC337 transistor could be used as a switch in the electronic circuit. When a small current flows into the base (B) terminal of the transistor, it allows a larger current to flow from the collector (C) to the emitter (E) terminal. The light source (which had inbuilt resistors) was connected to the collector terminal of the transistor and the emitter terminal of the transistor was connected to the ground. By connecting the base terminal to a pin on the microcontroller A small current could be Applied to the base terminal of the transistor to turn it on. By controlling the base current, the transistor could be turned on and off, thereby controlling the flow of current through the light source and creating the flashing effect.

One part of the Conga line algorithm could cause the light to flash in specific pulses, the other part of the algorithm could measure the light pulses from another source and calculate whether that source came from the leader. If it was found to be the leader the buggy would then follow it.

6. Results

After the sixth week of the Robotics section, the buggy was fully developed with the capability to run multiple different pieces of code. Figure 8 displays the completed buggy with all the components. This includes the four light tracking LDRs, the LED which shows if it is in lock and which helped distinguish the change in speed during the motor characterisation task, and the back-facing LED to pulse a light frequency for other buggies to lock onto. Using the light tracking software, the buggy was able to effectively follow a light, depending on the ambient light which could interfere if too bright. The robot would do well in dark environments due to the low ambient light. The buggy was able to implement the conga-line code to create a pulse with the backward-facing LED. The conga-line was never demonstrated due to the fact that it was not finished and tested in time. In the final stages of development, a cardboard square was used to block the front 2 LDRs off from one another. This prevented light from the source from hitting either hard left or right LDRs at the same time as their opposite frontal counterparts. In figure 9, a close-up of the wiring on the breadboard and Arduino is shown.

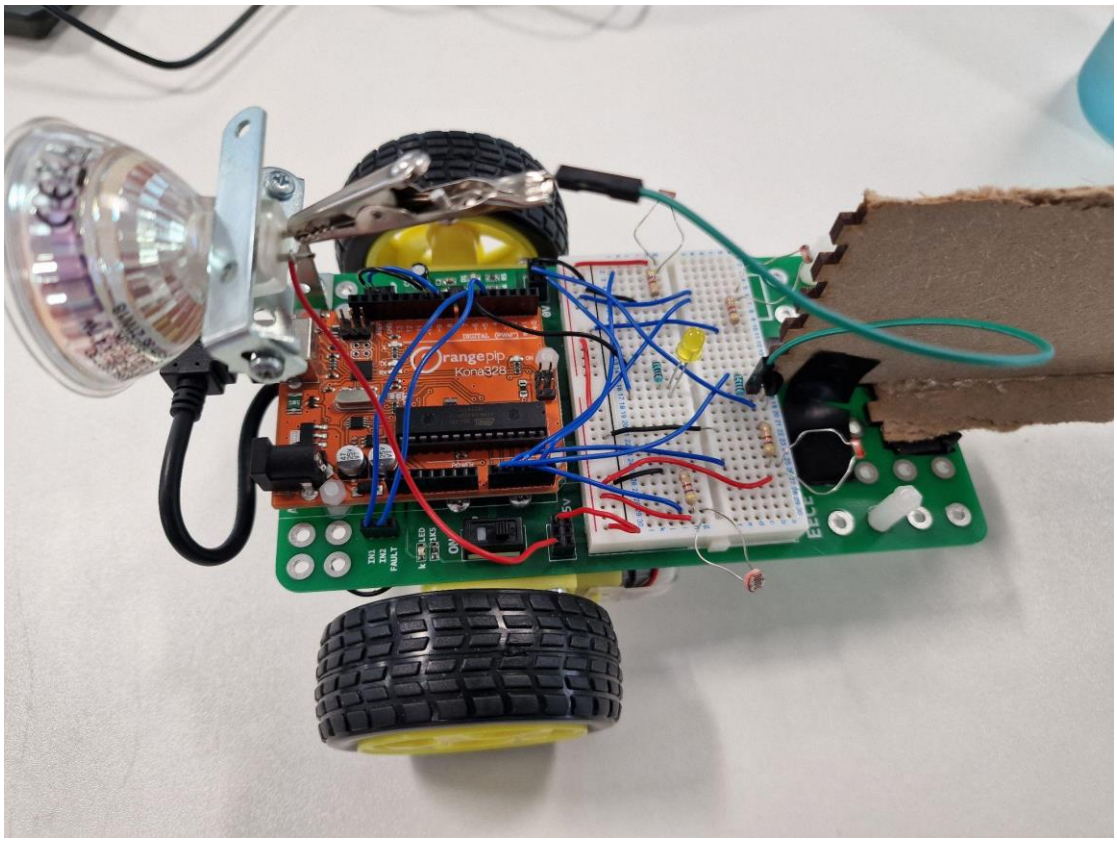

Figure 8: Final PWM tracking and PWM emitting robot.

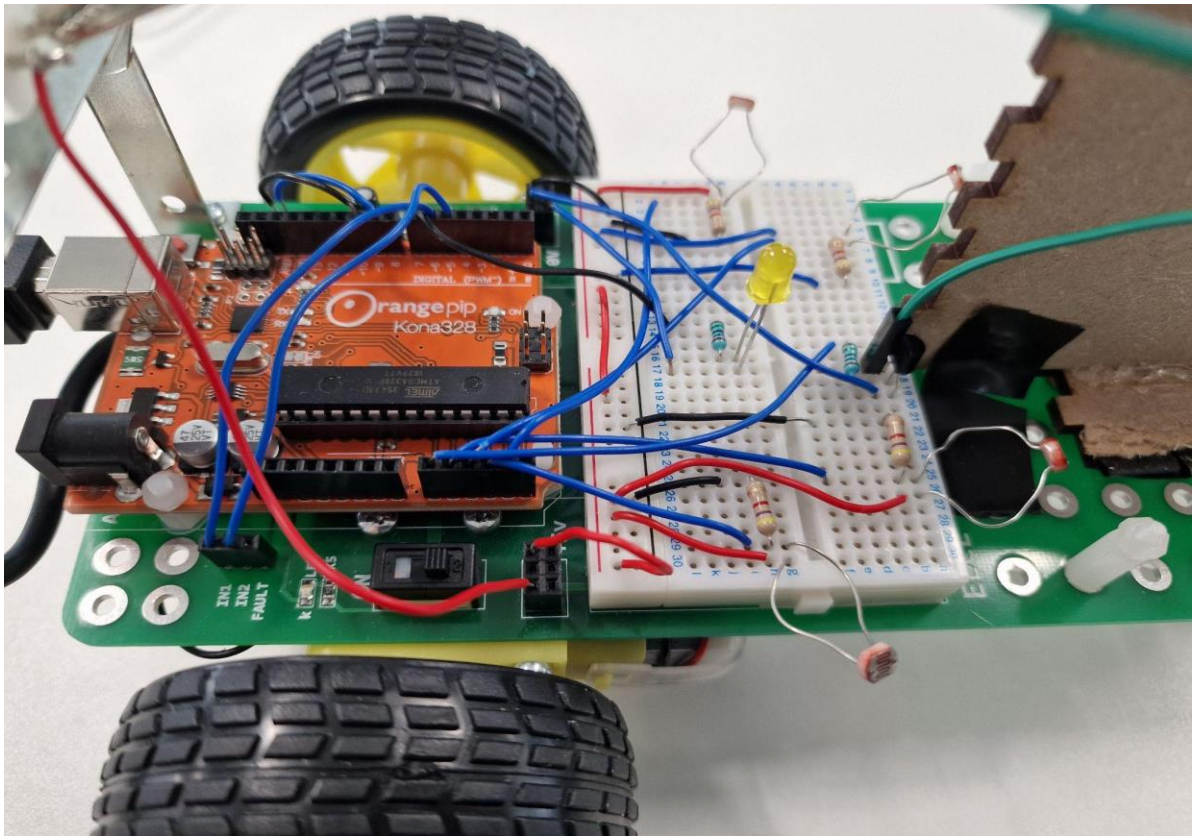

Figure 9: Final PWM tracking and PWM emitting robot. Circuit close-up.

7. Conclusion and Lessons Learned

In this robotics section the team learned how to edit the Arduino's programming in order to operate motors, LDRs and LEDs. To find appropriate parameters of operation, the motors were characterised, LDRs with resistors as voltage dividers were set up and the appropriate voltage measurements were taken, the robot light following code was written and tested as well as more advanced features to the buggy were added at the team's own discretion. In conclusion, the course greatly helped in teaching basic electrical engineering and robotics practices and helped build students confidence in assembling and utilising simple circuits.

8. References

https://store.arduino.cc/products/arduino-uno-rev3 -Arduino UNO board [1]

https://www.microchip.com/en-us/product/ATmega328P -AT mega microchip [2]

https://www.ti.com/product/DRV8833?qgpn=drv8833 -DRV8833 H-bridge [3]

https://www.mouser.com/datasheet/2/149/BC337-193546.pdf -Transistor datasheet [4]

https://media.diy.com/is/content/Kingfisher/5059340199603_FPD_ENpdf -LED datasheet [5]