

**EE475 Project Final Report**

**Using AI and Robotics To Entertain Cats**

Student: **Jack Storrie**

Supervisor: **Fredrik Nordvall Forsberg**

# Declaration

“I hereby declare that this work has not been submitted for any other degree/course at this University or any other institution and that, except where reference is made to the work of other authors, the material presented is original and entirely the result of my own work at the University of Strathclyde under the supervision of Dr Fredrik Nordvall Forsberg.”

# Abstract

This project aimed to create an intelligent robot that provides entertainment for cats, built using principles of robotics and AI. The product itself takes the form of a robotic toy that uses image recognition to make smart movements and evade the target (the cat) and stay within its boundary limits (avoiding the walls of the room, chairs and other objects on the floor that may obstruct movement). The product aims to be cheap, but effective with good functionality and a durable structure.

The product will is built around a Raspberry Pi Zero computer, which will be used to allow the AI functionality to be coded using Python. It will also contain motors used to run a set of wheels for the movement of the mouse. Other low-cost components such as the basic electrical components and either a small camera or ultrasonic sensor for object recognition will be used and packaged in a shell that represents the mouse as compactly as possible. By far the most important part of this project is to have as much robotic and AI functionality fully working as possible, but aesthetic look of the product must also be considered as well. A better looking product will be more appealing to the cat and will increase the amount of engagement that the cat has with the product.

Contents

[Declaration 1](#_Toc63702414)

[Abstract 1](#_Toc63702415)

[1. Introduction 3](#_Toc63702416)

[2. Project Goals 3](#_Toc63702417)

[3. Design Stage 4](#_Toc63702418)

[3.1 Hardware Component Selection 5](#_Toc63702419)

[References 9](#_Toc63702420)

# Introduction

This report will aim to give the reader a thorough understanding of this project, as well as the thought processes behind decisions made in the project, how the project progressed from beginning to end and key milestones within this timeline, as well as the technical background behind the components and techniques used in the project. An overview of the original project goals, and a review on to what degree these were achieved, will also be given.

# Project Goals

The aims of this project are split up into 3 distinct categories. Major goals, which are those that are essential for the completion of a working project, minor goals, which are not essential for an operational project but highly desirable for the completion of a high quality one, and a stretch goal that would take the project into more advanced technical areas.

The first major goal is to have the robot performing basic movement on its own across surfaces in all possible directions. Another is to implement basic AI object recognition. In its most basic form this would allow the robot to make informed movements as not to hit any obstacles, but it may not display very intelligent behaviour in terms of interacting with the cat or its general environment. The final major goal is to house all components as compactly as possible within the robot. The hardware needs protection and the robot itself has to be a reasonable size and have some sort of package or shell holding it together in order for it to be durable and structurally sound.

The first minor goal is to implement more advanced object recognition techniques to allow the robot to adapt to its environment. We want the robot to be able to make high quality informed decisions on its next movements by recognising particular objects in its vision. We also want the robot to be able to logically determine what it should do slightly ahead of time, and carry this out. The other minor goal is to create an aesthetically pleasing robot to the cat. We want the highest chance possible for the cat to be engaged with the robot, and completing this goal will ensure that happens.

The stretch goal of the project is to research and incorporate AI machine learning techniques into the robot so that it can display more intelligent behaviour by learning patterns of movement from cats each time it interacts with them. Completing this goal would allow the robot to have a database of knowledge, and use that in combination with what it detects in its current environment in order to make high quality decisions.

An evaluation of how effective these project goals were, and to what extent that the work carried out achieved them, will be included later in this report.

# Design Stage

The ‘design stage’ for this project involved considering what hardware, and software components should be used, as well as how the robot should be structured, in order to provide the best response to the project goals.

The first thing done in the consideration of the design, was to briefly research what other toys are already out there on the market for entertaining cats. This was done to give an idea of any patterns in the design characteristics of toys for cats which might be able to be applied to this project. Looking at available products, there a number of ‘catch the mouse’ style toys available on the market, such as the example shown in Figure 1, available from Australian retailer Mega Pet Warehouse[1]. This has a small model ‘mouse’ that can move around in a circular radius, moving under areas covered by plastic, and out again for the cat to catch it. This provided inspiration to make the design of this project a free-moving version of one of these catch the mouse style toys.



Figure 1: Catch the mouse style toy currently available on the market

## 3.1. Hardware Design & Component Selection

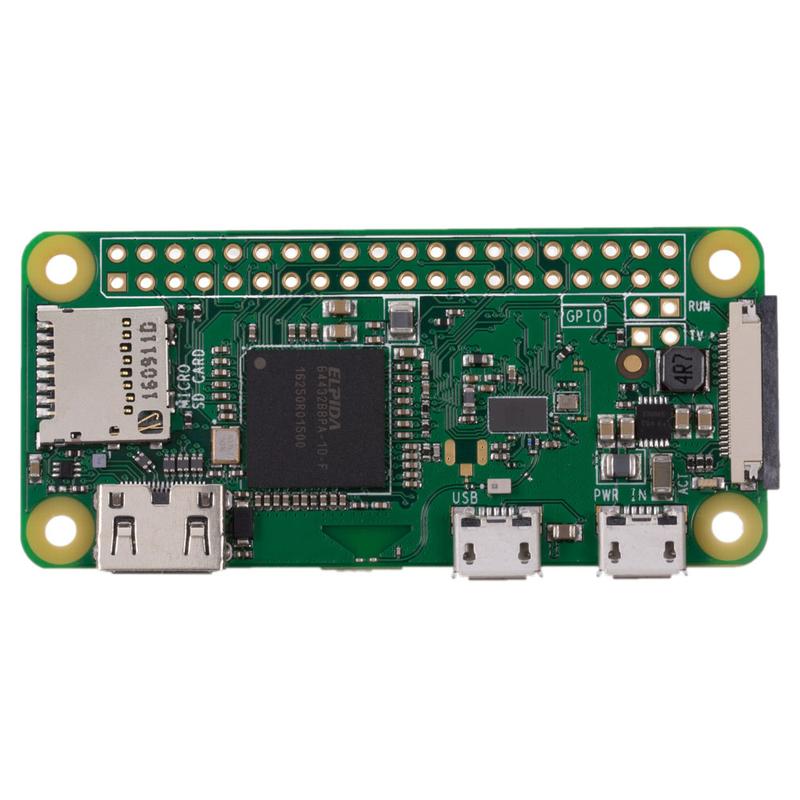
The next thing to consider was what components would be needed for the project. The main component in the project is the Raspberry Pi. In particular, the Raspberry Pi Zero W model (shown in Figure 2) was used, and this was provided to me by my supervisor. With this being provided from the start, the rest of the components would connect and build on top of this.

Figure 2: Raspberry Pi Zero W

In order for the robot to be able to perform image recognition, it needs a device to be able to provide visual input to it. The PiCamera, shown in Figure 3, is an attachment for Raspberry Pi that plugs directly into it via ribbon cable, with an adapter made specifically for Zero Series Raspberry Pis,, as the size of connector is slightly different. This is capable of capturing still image as well as video streams, and it is the device that allows the robot to perform object detection and image recognition.

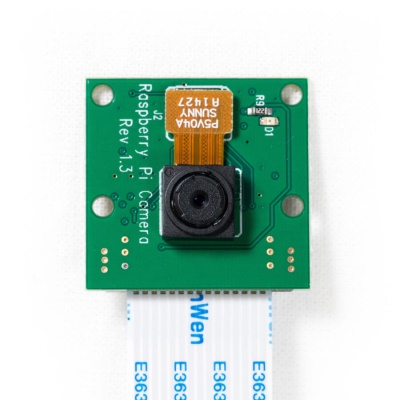


Figure 3: PiCamera attachment

Next, the robot needs components that will allow it to move. Intuitively, this is done through 2 motor-and-wheel sets. However, these alone wired to the Raspberry Pi don’t allow for the highest degree of control. Adding an H298 H-Bridge Motor Controller provides this extra control by allowing both motors to be controlled separately. The motors are wired to the H-Bridge, then this is wired to the GPIO pins on the Raspberry PI, allowing both motors to be commanded a single GPIO pin, and have different signals sent to them if need be. A diagram of where 2 motors connect to an H-Bridge is shown in Figure 4. A ball castor was also sourced, this is a cased metal ball that is designed to be placed between the 2 wheels of the robot. The reason for this, is to aid with friction, and easier movement across surfaces. As the robot is likely to be used on a surface such as carpet, which could potentially cause a lot of friction to the robot wheels’ rubber tires, the ball castor is a good addition to the project.

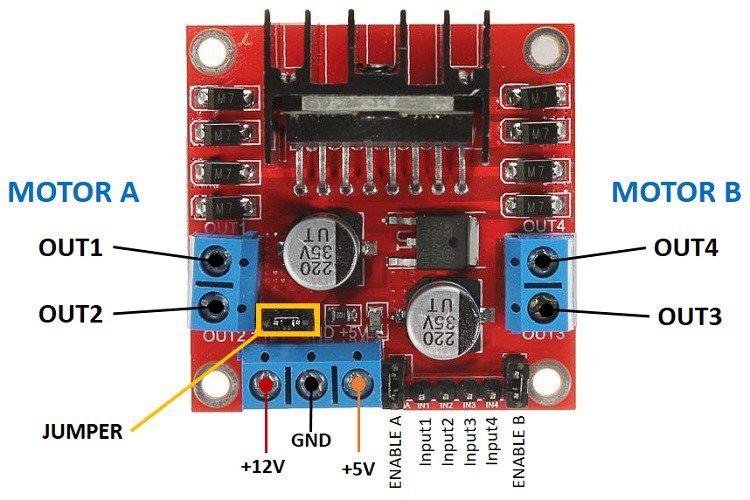


Figure 4: Connections on an H-Bridge Motor Controller

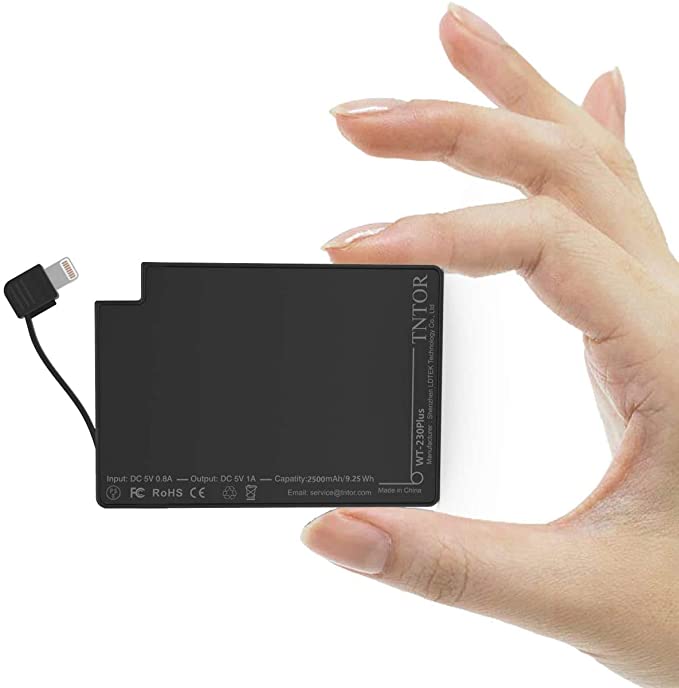
Since this is to be a free-standing project that can move by itself, the Raspberry Pi cannot be attached to a computer. It must therefore have some sort of external power source that is compact and portable enough to go along with the robot. For this purpose, a TNTOR WT-230Plus power pack (shown in Figure 5) is being used. This simply plugs into the PWR IN input of the Raspberry Pi, and provides its power while the robot is running.

Figure 5: TNTOR WT-230Plus Power Pack

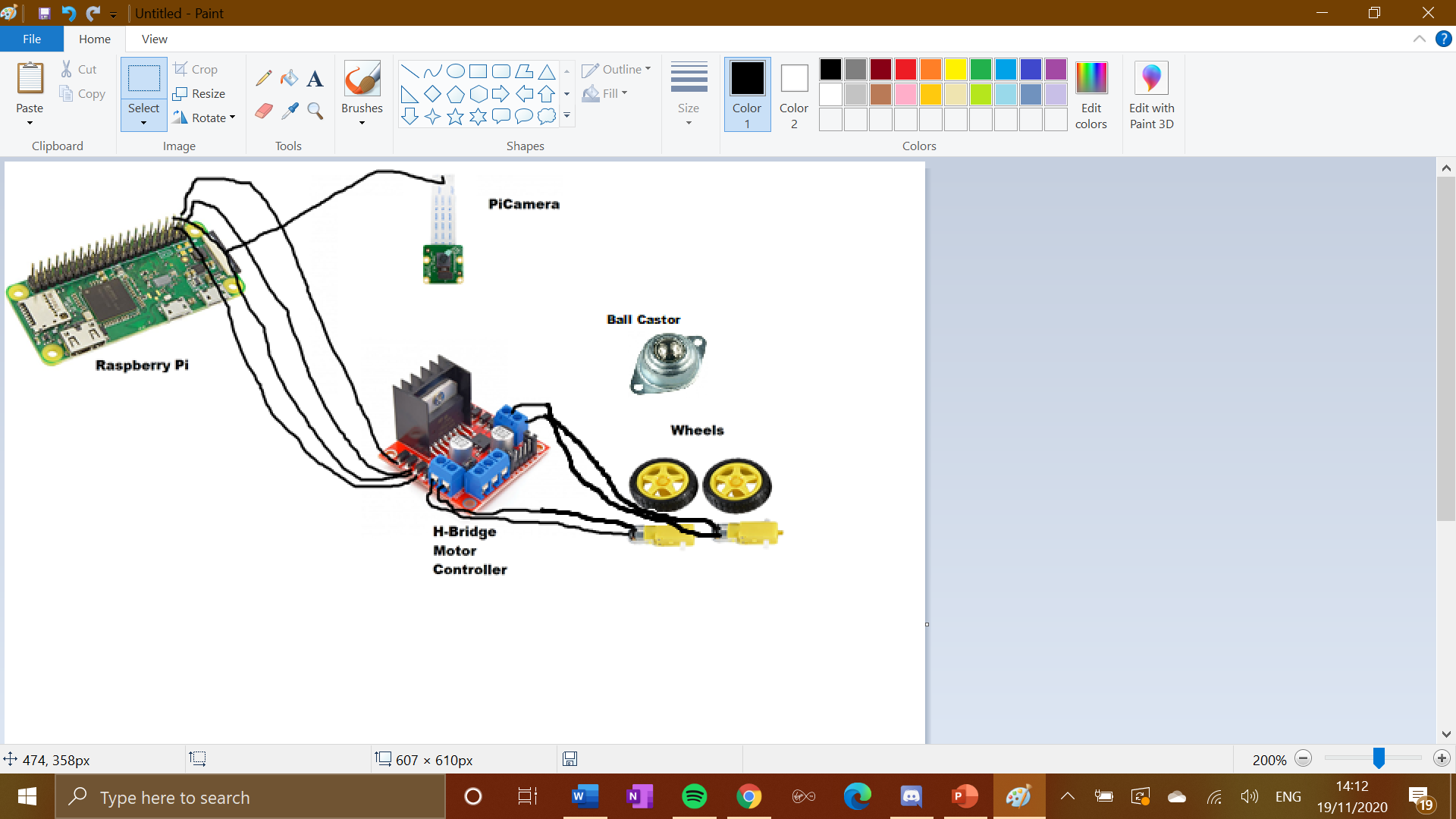
These were all the components that I chose for the original design of the robot’s hardware. A diagram I produced of how these components would be connected together roughly is shown in Figure 6. This highlights how the flow of code instructions goes from the Raspberry Pi’s GPIO pins, directly to the Pi Camera, and also through the H-Bridge which is then wired to the motors, which subsequently turn the wheels.

Figure 6: Original hardware design diagram

I later reconsidered the hardware design to include an ultrasonic range sensor. This is something I had originally considered but not put into the design, However, with the camera only seeing in one direction, the robot’s ability to make intelligent decisions would be extremely limited by ‘tunnel vision’. Therefore an ultrasonic range sensor was added to the original design. It is a standalone attachment, that is wired directly to the GPIO Pins on the Raspberry Pi. This allows the robot to have some sense of depth perception, to let it know if it is approaching any obstacles in any direction, rather than just in the direction that the camera is facing.

After the ultrasonic range sensor was added to the design, I believed I had included all the components I required in order to allow me to build a robot that is capable of making intelligent decisions.

## 3.2. Software Considerations

A stipulation of the project was that the python programming language is used to code the robot’s software. Python is an ideal language for this project as it handles AI functionality very well.

The main consideration in the programming of the robot was how to turn what the robot sees through the camera into instructions on where it should move. The GpioZero Python library allows hardware to be associated in code with specific GPIO pins on the Raspberry Pi. This is utilised in this project to allow signals to be sent to the wheels through the motors via the H-Bridge. A PiCamera Python library also exists, which helps define the camera’s functionality and allows it to be handled directly within the robot’s Python code.

When researching object recognition techniques, I discovered an article covering an OpenCV deep learning object recognition library for Python[2]. This has many pre-defined objects in its database that it can instantly classify given a stream of visual input. Some of these are particularly helpful in this application, including “cat” which will be the most essential. Other objects in the OpenCV database that are helpful include “chair”, “table”, “door”. What makes this so useful is that as well as classifying objects that it recognises, it gives them a “box” which is a square area roughly the size of the object with a small margin of error. This allows our robot to then move towards avoid a certain object by tracking its box. The way in which this library classifies an object is by comparing any objects it detects in the frame, against its own object database. Any matches are given a degree of confidence, and by employing minimum threshold on the degree of confidence of a detection, we can decide whether or not to take any action according to that object.

## 3.3. Pseudocode Design

After researching how the OpenCV database could be applied to this project, I decided to create a preliminary pseudocode software design to outline the operations and possible functions I believed would be necessary within the program. I decided that visual input should be taken by the PiCamera using a video stream loop, similar to that of CCTV. Taking still images at regular intervals was another possible approach, but it would be difficult to take the image ,analyse it, and choose an action before the state of the robot’s field of view had significantly changed. Using video allows for decisions to be made in real time. Using a CCTV-style video loop instead of constantly recording saves storage space, and allows the program to reset its state at regular time intervals (e.g. the start of each video loop). I decided that once an object is recognised, it should be stored in a temporary database for the remainder of this video stream loop. Once the object detection algorithm has run its course, the objects detected in frame and their location is then passed to the movement algorithm, which will use this information to determine what the most logical next movement step for the robot should be, and sends appropriate signals to the motors for this.

The pseudocode design is shown below:

define camera and motors as variables

main\_function():

camera\_setup()

while(running):

camera\_processing(video\_stream)

movement\_algrorithm(detections)

return

camera\_setup():

activate\_camera

set\_up\_video\_stream(definelengthofloop)

start\_video\_stream

return video\_stream

camera\_processing(video\_stream):

define variable for objects detected called detections

define variable for locations of detected objects called coordinates

scan frame for objects

add objects detected to detections

find location of detections in frame

add locations to coordinates

return detections, coordinates

movement\_algorithm(detections):

scan through object locations

decide on next direction signal to motors

send directional signal to motors

return

# Implementation Stage

## 4.1 Initial Setup

## 4.2 Individual Component Testing

The next stage was then writing and performing tests for the major peripherals. Testing out the individual components to make sure they can perform at full functionality is important, as it rules out a large number of potential issues when it comes to debugging at a later stage, and when all components are connected together.

### 4.1.1 Testing the Camera

For the camera, 3 basic test functions were written to ensure full operation. The first was to capture a still image, outputting an image file names ‘foo.jpg’. The code for this test can be seen in Figure 7. I then ran the test and the captured image can be seen in Figure 8. The camera was facing upwards at the time so it only captures my wardrobe and ceiling, as well as the smaller object of the smoke alarm. However it still shows that the camera captures image with decent enough quality to make out the objects that are in its view.

Graphical user interface, text, application, email

Description automatically generated

Figure : Code for testing capturing a still image

A dark room with a window

Description automatically generated with low confidence

Figure : Image captured during test

Graphical user interface, text, application, email

Description automatically generatedThe next test was to capture a short video. The picamera captures video in the .h264 file format. The code for this test can be seen in Figure 9.

Figure : Fixed-length video capture test

The final test of the camera, and the most crucial one for this project, was to test the camera’s ability to capture circular stream video. The code for this test can be seen in Figure 10. What happens here is a ring-buffer is used to restart the stream and overwrite the previous loop. The last n seconds of captured stream video is held on the Raspberry Pi’s memory, with n being determined by the size of ring buffer Graphical user interface, text, application, email

Description automatically generatedimposed on the stream as well as the bitrate of the video.

Figure : Stream test code

The most attention was paid to the looping stream video as this is the method that was determined to be best for the project to aid the movement of the robot during the design stage.

### 4.1.2 Testing the Motors

For the motors, functions were written and carried out for varying tests of movement for the motors and wheels, namely, the motor action required for moving in a square pattern, and moving in a continuous circle. The aim was to test the ability of the motors individually before the project moves on to a more advanced stage. These two tests demonstrated that the motors and wheels could turn in all required directions.

First was the square movement test, the motors were first instructed to motion forward, before stopping, then instructed to move to the right 3 times, completing the square. The code for this test can be seen in Figure 11

Graphical user interface, text, application, email

Description automatically generated

Figure : Code for square movement test

The circular movement test first starts motioning the motors forward similarly to the square movement test, but then enters a loop of steering right continuously in the ain of producing a smooth circle with the wheels. The code for this test can be seen in Figure 12.

Graphical user interface, text, application, email

Description automatically generated

Figure : Code for circular movement test

On discovering that the main components of the robot were fully operational as they should be, the next stage of the project was to start developing the software structure outlined in the pseudocode design that will contain the movement and object recognition algorithms and all other code for the operation of the robot.

# References

[1] <https://shop.megapet.com.au/catch-the-mouse-motion-cat-toy-0639737307900-25941/>

[2] https://www.pyimagesearch.com/2017/10/16/raspberry-pi-deep-learning-object-detection-with-opencv/