Boswin 3.0 - navigation investigation.

Investigation into the receptiveness of ultrasonic sensitivity with regards to spatial awareness.

Analysis:

# Project background:

The project aim is to develop upon a prior project consisting of a combination of artificial intelligence and engineering to form a free roaming bipedal walking robot.

Prior endeavours into autonomous surveillance were focused upon manoeuvrability focusing on the use of lightweight materials to have increased flexibility from the several 9-gram servo motors used. However, on this further endeavour the use of an Arduino Nano v 3.0 would drastically reduce the mass compared to the previously used Arduino Uno which had a considerable weight increase. Using lighter materials for the device's composition would allow for the inclusion of further expansion with the use of sensors to comprehend its situation and continue under suitable means.

# Prior project research:

As for mentioned I have participated in previously investigation into navigational systems, the project focused on the power distribution as well as the conjunction of four servos performing in harmony. The design was primitive at best and had no capability for true guidance and performing meaningful investigations as intended from the new design. However, the previous design provided keen information with regards to light weight and custom manufacturing of components. This has allowed for a higher success and completion of the design since adapting conditions to suit preferences has led to a much higher precision and accuracy of the hardware’s construction to provide minimal external error to allow focus on the functionality of the software.

The original concept spiraled from an open source project under the title “Arduino Otto” the robot was very similar and can be found at “ <https://create.arduino.cc/projecthub/cparrapa/otto-diy-build-your-own-robot-in-one-hour-5f2a1c> ” which is where the original versions of the devices casing can be found, however in comparison to my own version, my own functioned using a much higher powered Arduino Uno instead of the more compact Nano board which the new project will use furthermore my own device operated with the ability to function wired or internally.

Since hardware concerns had been addressed in previous renditions it allows for there to be focus on software the code can be re made with a much more condensed format in comparison to the previous project’s excessive nature allowing for more freedom to implement new features as there is reduced occupancy of the devices internal memory.

Furthermore, the previous project was unable to actual navigate through its own functionality but was essential blind to its surroundings. To improve this the choice to add two onboard sensors, one being an ultrasonic sensor which is explained in further depth in the segment “Estimations of the effectiveness of ultrasonic guidance” which goes into depth about the projection of wavelengths and how returned data is used to aid the devices operation.

Secondly there will be the inclusion of a tilt switch which will only be used as a secondary measure to rectify the device in case of an error in its functionality. The switch is operated via a small ball contained within a tube which when value is “HIGH” it means that the ball is creating contact with the two small connectors at the bottom of the cylinder completing the circuit as demonstrated in fig.1.

(fig.1) The left most image represents when the switch is right way up and the switch is triggered on due to the contact between the two nodes and the ball segment.

The right most image demonstrates when the switch is not right way up and the movement of the ball Is away from the two nodes. This would leave the sensor off.

Prior investigations had demonstrated that the tilt switch is a useful device however not entirely reliable as a main source of determining the position of a device. This is essentially why the sensor is delegated to a secondary sensor in the investigation.

# Hardware characteristics:

The device would consist of a light weight 3d printed casing and operating via an Arduino Nano 3.0 and input/output shield. The device, capable of operating up to 12 servo motors, would be used to operate 6 servo motors as well as an ultrasonic proximity sensor and a tilt switch to provide the micro controller the essential spatial awareness in order to successfully navigate obstacles.

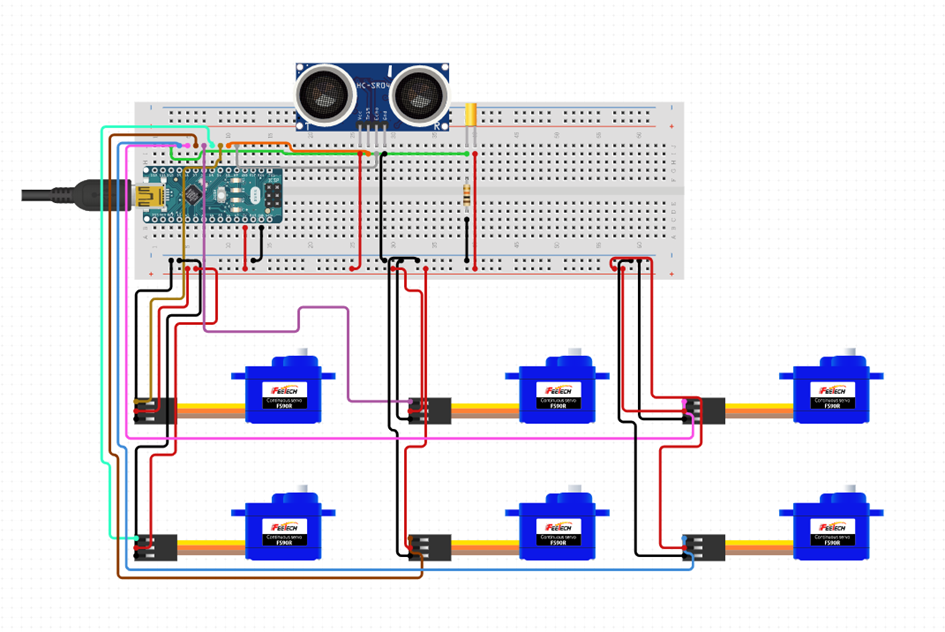
The method in which these will be implemented is demonstrated in fig.2.

The micro controller would be capable of power distribution for the devices as well as relaying information from the sensory devices to provide an appropriate action to deal with the information in order to continue the free roaming ability with regards to both terrain and obstacles.

The device should hopefully provide insight into an investigation of ultrasonic feedback and consideration of gradients when negotiating an environment.

For maximum use of feedback the device will relay information to Arduino providing a real time output allowing for analysis of data while in use, however the device will also be able to operate via a micro USB power supply for further roaming ranges without the requirement of a physical connection.

By having a mobile device, the range of ultrasonic surveillance would be much larger due to the independent autonomy of the device.



(fig.2) general overview of the internal hardware wiring of the device.

The small blue segments are the six servo motors to make the device move by fluctuating their position in a range of 0-180 degrees, this allows for six degrees of movement as each servo provides a single articulation point.

The section which has two large circular components is the ultrasonic sensor for the guidance of the device, which its functionality is further explained in the section” Estimations of the effectiveness of ultrasonic guidance”.

The small golden device contained within the red circle annotation is the secondary sensor, also known as the tilt switch, which is used for determining the gradient of the device.

The green circuit in which the mini USB type b is connected to is the microcontroller for the device which manages all the actual computing and power distribution required.

# System requirements:

The intended operation is to use the information relayed from the ultrasonic sensor in order to decide between a group of potential movements.

Where suitable the substitution of 3rd party libraries for custom written classes however I intend on using the servo library due to the extensive depth of the library and the excessiveness in re-working the library for no functional gain.

**These will consist of:**

Move forward, take several steps forward covering a set distance and if at any point during the cycle a given boundary is breached (moving too close to an object) it will halt and resume a neutral stance to await a sequence of next moves, most likely made up of the turn left function, turn right function and the walk forward function to resume its exploration after finding an obstacle.

Turn left, will be a given set of movements to increment its movement in an anti-clockwise motion by 90 degrees, allowing the analysis of if the path is clear for a given area.

Turn right, would operate in a similar manor to that of the turn left function, but will proceed through a 90-degree movement clockwise.

Move backwards, this will most likely be one of the least used movements due to the lack of guidance when reversing, however it would proceed in a similar manor to move forward, but to a shorter distance due to the given uncertainty.

Through these functions the device should have the means to negotiate its local environment. The device would be completely dependent on the ultrasonic sensor for its guidance; however, the device will also contain a tilt switch which operates based on the relation of the angle of the device to the sensor. This will be used to determine if the device had any issue and had fallen over. Once indicated the device would be able to provide a suitable alarm to the user then end its operation to conserve its power source.

# Estimations of the effectiveness of ultrasonic guidance:

The ultrasonic sensors operation would mimic that seen throughout nature such as bats. Due to the sensor being completely isolated from being obscured it allows for the navigation under any condition due to lack of dependency on light and other conditions the device is much more open to further free roaming opportunities.

However. the ultrasonic dependency may lead rise to issues as a result of the density of air which could cause obscuring of the wave lengths.

This would occur due to the reliance on the constant given for speed of sound as averages of 343 m/s typically occur in 20°C conditions and due to the fluctuation of air density this may alter causing degrees of error in the distance measuring.

One of two approaches could be considered when facing this problem which one is to provide an interface in which the temperature can be given as an input however this would require the device to be physically tethered to a computer reducing the free roaming capabilities the alternative is to assume an average which has been assumed to be 340 m/s. this can then be used by the sensor to calculate the relative distance of an object. By using the equation time= distance/speed it can be assumed that the time in which the projected 40,000 hertz wave takes to return to the sensor’s receptors. Then with the assumed constant of the speed of sound the desired calculation is simple being in the form distance = time\*0.034/2.

The second option would prove most effective to the current needs as the devices potential fluctuation of error is only that of 0.001 metre per second therefore it is unnecessary to include the alterations as the overall changes in values would be that of millimetres and have no actual effect on the robots operation and would also conserve the limited storage space of the Arduino Nano micro controller.

This is summarised in the diagram fig.3.

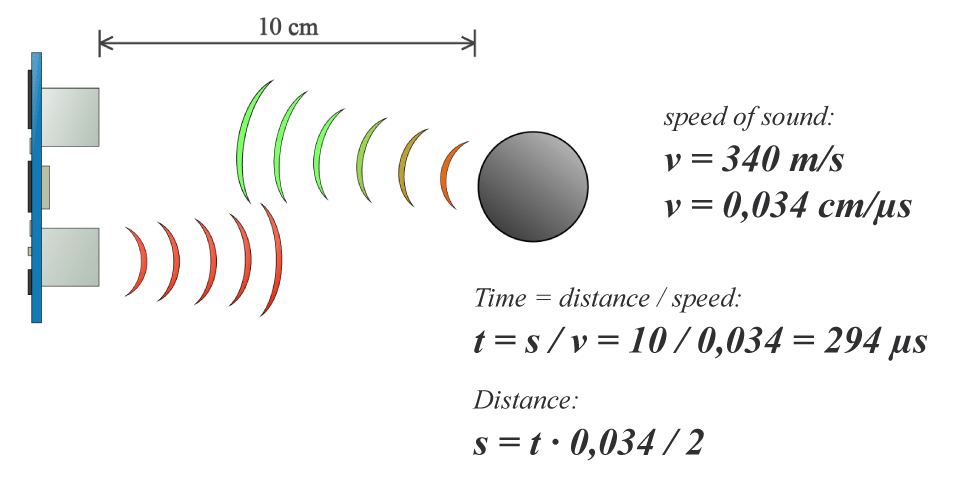


Fig.3 concluding the method in which ultrasonic sensors function.

Sourced from <https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/>

**Feedback loop analysis:**

The device is intended to operate through a series of checks and conditions in order to determine if the data is consistent enough to allow the device to progress its investigation.

Fig.4 concerns a diagram which provides the format in which feedback checks and conditions are received then used to find an appropriate course of action.

The flowchart outlines the eventually required feedback loop processing both the ultrasonic guidance information and the tilt sensor correction information. This also covers the eventual output or reaction to the information in the form of the servo motor movement.

The chart provides means to check for several issues as well as having several means to resolve the given issue in order to continue its investigation.

The flowchart provides four routes once an error has occurred.

The first route is the device to step backwards to provide space for further action to be taken.

The second route implies that the robot turns right by 90 degrees clockwise and checks if the area is clear to continue its roaming.

The third route provides a turn left movement from a right position of 90 degrees clockwise to 270 degrees by a movement of 180 degrees anti clockwise.

The fourth potential route proceeds to check if the device is up right is this is not the case the device will attempt to correct this issue through its use of servo 5 and 6.

Unfortunately, if all four courses of action fail the device is no longer able to continue its investigation therefore will halt all checks until the device is rectified and reset by the Arduino Nano's built in reset function.

However, the possibility of actual failing to rectify any issues after these further attempts is very minimal and would be unlikely to occur within regular operation.

The chart contains two check variables one simply being called “check” and the other being “tip check”. The variable check is used for the program to count which of the previously mentioned procedures are required based on the number of loops.

Tip check is used to analyse if the correction gradient code using servos 5 and 6 has been successful and if they have not the tip check variable indicates that the program should end due to no potential further course of action.

A close up of a piece of paper

Description automatically generated

Fig.4 feedback loop flowchart, featuring the implementation of a given feedback from both the ultrasonic sensor and tilt switch.

The flowchart outlines the alternatives and procedures to deal with and manage issues to arise, if all else is to fail in the devices operation a third party is to intervein to rectify the issue, however the likely hood of that happening should be extremely minimal and is only represented to represent what courses of action would be taken before this stage occurs.

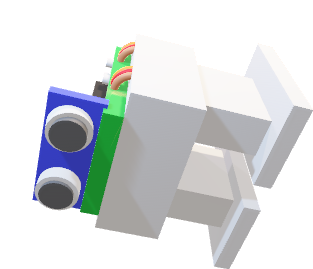
**Device production and hardware implementation:**

The devices hardware has also severely altered to that of the prior investigations by the inclusion of sensors, onboard power, and two extra servos this has led to somewhat of a redesign to prior projects in order to accommodate the additions.

First of the adaptations is to provide mounting points for which the extra servos (servo 5 and servo 6) to allow for the addition of the arms as seen in fig.7 whereas this was not prevalent on the prior renditions as seen in fig.5 and fig.6. furthermore, the inclusion of ultrasonic sensors was featured on both fig.5 and fig.6 however, the last of the three will actual be used as a functional component, this was considered previously however was not included due to power distribution issues.

In further adaptations the choice to include a specialised breadboard would allow the Boswin 3.0 device to distribute power in a more concise method to that of Boswin 2.0 which had an unreliable soldered connection which left the device being unable to be altered and varied when required. However, using the Arduino Nano's break out board and non-permanent solution it allows for the device’s potential expansion over a longer term.

Finally design alterations were made to the 3-D printed casing allowing for easier production and improved access to the servo motors for replacements and upgrades over the course of its development. Further alterations were made in the main body to allow for increased access and internal storage which will continue with improvements made by Boswin 2.0 and onwards as Boswin 1.0 was only partially cased leaving an exposed microcontroller and circuitry.

****

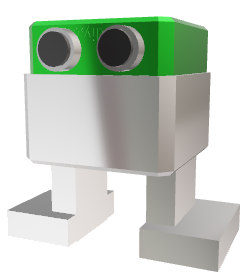
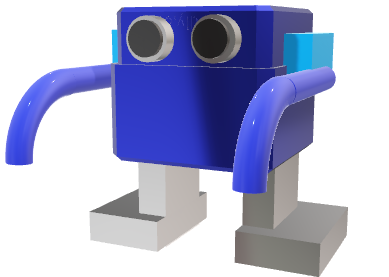
****

Fig.5 Boswin 1.0. Fig.6 Boswin 2.0. Fig.7 Boswin 3.0.

**Objectives:**

From the conclusion of the analysis the following is a list of required objectives and requirements which are required to enable a successful investigation.

* The device should be capable of processing information and locating the suitable data in order to confirm its next function.
* It should be able to operate completely independently without the use of a computer except from a user recording data feedback.
* The device should be able to be easily altered and have accessible maintenance to components such as sensors and the microcontroller board.
* The device should be capable of alerting the user to a situation which is not rectifiable.
* It should include a fail-safe function for when no further actions can be taken.
* The device should have an operational independent lifetime of up to ten minutes as a minimum.
* The device should be able to output error messages, data inputs and any necessary information to the user when connected to the Arduino ide serial monitor.
* The device should be able to recognize situations where no further action can be taken and halt operation.
* The device should have appropriate conditions to verify the condition of the device.
* The device should have a suitable durability to deal with a variety of scenarios which it may encounter.
* The device should be full contained as to prevent issues with regards to the hardware's integrity.

Design

## System overview:

For this investigation, two factors should be considered when evaluating the design, those being the hardware and software components.

**Hardware:**

Firstly, focus was based on regards to the hardware foundations, this was chosen as the start point as the software would have to be designed around the device’s hardware factors.

The overall design is based on a casing made of P.L.A plastic a non-toxic biodegradable substance chosen for its ease of alteration for additional features and components, the plastic is non-conductive and the most suitable casing to cope with the scenarios and conditions at hand. Furthermore, the casing has been altered to allow for fewer unique parts to make repairs and substitution more accessible.

Once the casing was made installation of hardware was evaluated, the chosen micro controller was based on an Arduino Nano as formally mentioned due to its significant weight decrease to the previously used Arduino Uno on the prior Boswin 1.0 and 2.0 models. Once the micro controller was chosen finding the appropriate components was the next task.

The device was initially given 6 continuous 9 gram servo motors, however it would prove that this was insufficient therefore alternatives had to be considered, due to a deterrent to minimize alterations to the main chassis of device, I had sourced 6 non continuous metal geared servo motors which replaced the previously plastic gearbox mechanism to the mentioned metal gears, by doing this issues of the smaller plastic gears shredding the gear teeth were eradicated, allowing for more sustainable use of the device.

Further inclusions were of an ultrasonic sensor and a tilt switch.

The devices standard means of guidance would be the ultrasonic sensor, the device itself is of minimal complexity however is prone to some error where the sensor constantly measures the value of zero, this has been accounted for within the software section, however functionally had no significant alterations.

The device would also receive a less frequent check of the imbedded tilt switch which operates as a secondary input verifying the orientation of the device, the original intention was to use this as an indicator to activate the self-righting equipment however the used servo motors are incapable of providing the required force to rectify any positional errors of the device, so as a result of its supposed redundancy the device will use the information of the sensor to send a sort of error message to inform the user or a third party that the device is in need of assistance, in order to continue onwards.

The remaining inclusions were all connective wiring and a rechargeable battery which can operate the device for five to ten minutes as a result of the power drain of the device, however the device is capable of operating while wired which would also provide a means for information to be relayed to the serial monitor on Arduino, the information will be used throughout testing for the purpose of receiving suitable feedback to develop the device further.

The devices inclusion of a tilt switch was included however errors with hardware and lack of availability has resulted in an alternative option of using a micro switch to be triggered when the device has fallen over, this results in the devices software to exit to prevent any damage of the device from improper operation.

In conclusion for the hardware, most of the design is a resemblance of the previously included wiring diagram, with minor alterations to fit within the size constraints.

In total all hardware used were:

* 1x Arduino Nano micro controller
* 6x 9 grams metal gearing micro servo motors
* 1x micro switch
* 1x ultrasonic sensor
* 2x heavy duty single servo horns
* 2x heavy duty linear servo horns
* 2x heavy duty quad servo horns
* 1x lithium polymer 2 cells 7.4-volt battery
* 1x custom fitted breadboard
* Approximately 120 cables
* 2x feet adaptors
* 2x leg adaptors
* 2x arm attachments
* 1x lower casing
* 1x upper casing

**Software:**

The software design was implemented through several attempts of working on individual functions and then bringing them together in order to work.

Primary focus was placed on the implementation of the ultrasonic sensor, this proved to be one of the simpler steps where the main difficulty was ruling out erroneous values.

Once the function of the ultrasonic sensor was established the addition of the servo functions was to be added, which related the output of the ultrasonic sensor to the manipulation of the servo motors.

In order to do this set means of which certain components could be transmitted to and from had to be established, these were recognized as pin numbers which can be chosen from 1-13 as that is the range which the Arduino board is capable of.

The code implementation was mainly a method of trial and error as unlike with regular programming where the program can be run within the editor of choice, all testing and corrections had to be done simultaneously.

The coding section initially was based on the operation of individual functions, those being,

* Ultrasonic sensor functionality
* Servo motor conjunction use
* Feedback loops of operation
* Walk cycle development
* Logical progression
* Failsafe procedures.

Firstly, the ultrasonic sensor implementation required the previously mentioned formula from the analysis section to be utilized, this provided the highest degree of accuracy possible within reason as further values beyond 343 metres per second (as of 20 degrees Celsius at sea level) would cause indecisiveness of distances.

As a result of the equation for distance being minimal and with integer results the data is managed as so, the data is then outputted to the serial monitor within the Arduino IDE in order to provide results for the user which was of particular use within development stages.

When investigating the implementation of multiple servo motors being used in conjunction, to do this a function was implemented called “servo collect set” using an input from an array to be allocated to each of the six servo motors this allowed the numerical values of the servo motors being set without having to allocate each servo motor, every time the servo values are to be configured.

The values in figure 8 of s1\_set through to s6\_ set is the notation used for the servo motors, most commonly the values are set to a constant of 90, which has been declared under “pos” then for further manipulation of the motors ranges of pos-90 to pos+90 as that is the maximum range of the motors.

Furthermore, in figure 8 the use of consecutive servo.write functions allow for the most synchronized setting of motor values to extremely miniscule latencies between the validation of the values.

void servo\_collect\_set(int s1\_set, int s2\_set, int s3\_set, int s4\_set, int s5\_set, int s6\_set){

s1.write(s1\_set);

s2. write(s2\_set);

s3.write(s3\_set);

s4.write(s4\_set);

s5.write(s5\_set);

s6.write(s6\_set);

}

After the verified function of the previous two components the next stage is to integrate the feedback and results of ultrasonic sensor then using that to take an appropriate action in consequence to the data provided, this was briefly explored in figure 4 however is developed as the approximated values had to be refined to provide more realistic actions to account for the devices hardware factors.

Once the conjunction between feedback has been established refinement of the devices walk cycle will allow for more substantial movements of the device, to allow for more reliable navigation, the device is to utilize the arms which were originally intended to self-correct the device as counter weights to limit the possibility of catastrophic errors.

Using the newly optimized walk cycles, the functions to allow the device to negotiate obstacles need to be developed, this is accomplished by analyzing 90-degree areas at a time and validating if the new potential area is viable for continued progression of the device.

Finally, to reduce damages and potential issues with a navigation a set of failsafe protocols, the initial goal was the inclusion of the tilt sensor, however after implementation and development the hardware was flawed and required replacement, therefore a substitution of a micro switch was chosen which is triggered when the device has fallen over which then uses the exit(0) function therefore ending the devices operation, the second of the two failsafe's is included when the device has no logical progression the device ends its function, this helps to highlight when the device has found a flaw of the implementation of the devices ultrasonic detection.

**Documented design, hardware:**

The device began with the hardware orientated design, as to the completion of the device the alterations to hardware has been substantial.

Initially the device operated using an Arduino Nano connected to a motor driver board, however this resulted in power distribution issues therefore a standard Arduino Nano board was chosen.

Furthermore, issues were raised with the power supply which was initially a lithium ion power supply, however this was temperamental and had severely limiting weight and longevity factors, therefore the previously mentioned lithium polymer battery was chosen, which has significant weight decrease from approximately 20 grams to only 32.2 grams proving very beneficial to the devices operation. Along with the devices switch to metal geared servo motors, the devices performance has been greatly improved.

Internally, the device has extreme space limitations due to the required connections inside as shown in figure 9.

For this reason, the device has all technical components contained inside as can be seen in figure 10.

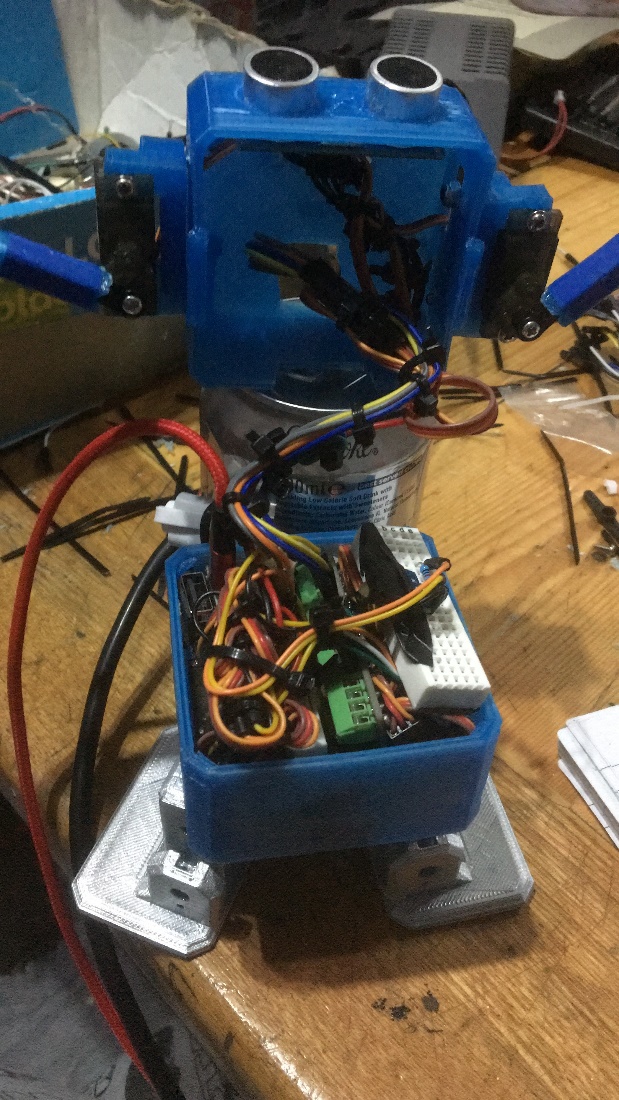
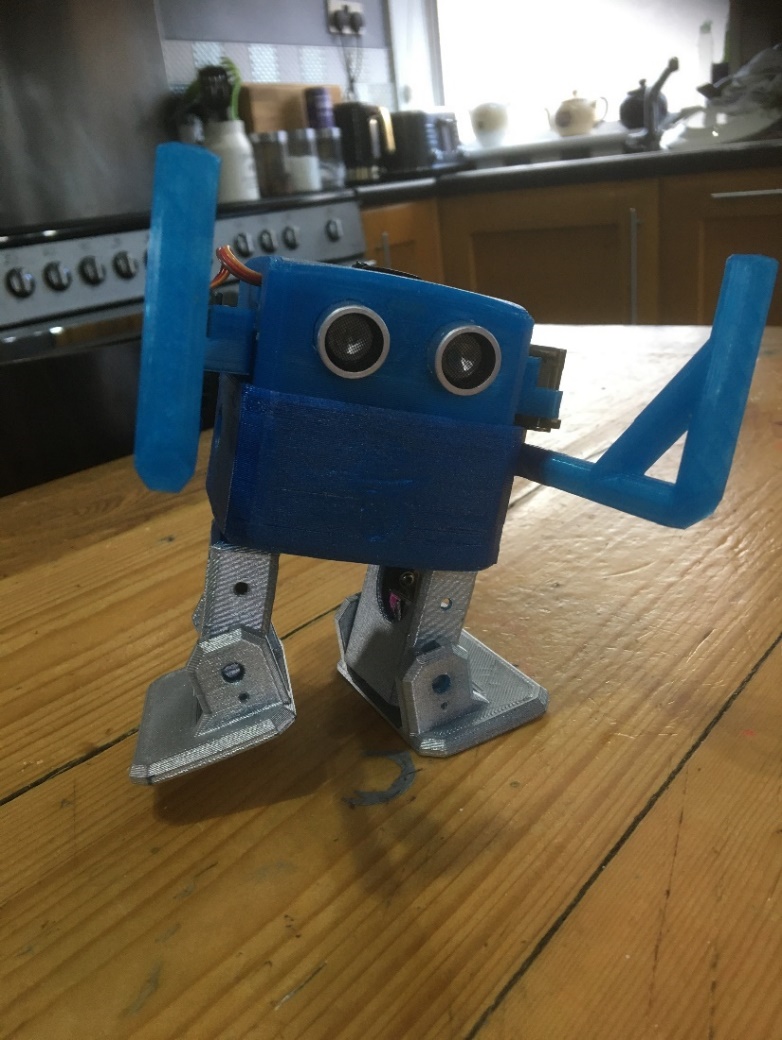


Fig 9, internal wiring and components. fig 10, assembled device.

The device as previously mentioned is encased in a 3d printed chassis, preventing risk of electrical damage due to its prevalent lack of conductivity, making the device much more durable and more versatile for experimentation.

As many as possible of the connections were hard wired to prevent the potential of poor or incorrect contacts, this also adds to the as fore mentioned durability, reducing the necessity to repair or re connect components.

Further developments were made with regards to the device's internal components and have allowed for increased access to its internal components. After the previously mentioned hardware variations, due to less fixed position attributes flexibility of the devices components is much more accessible, as seen in figure 11.

As well as the addition of a brass plate in order to act as an external counterweight has allowed much more reliable motion when in transit, as seen in figure 12.

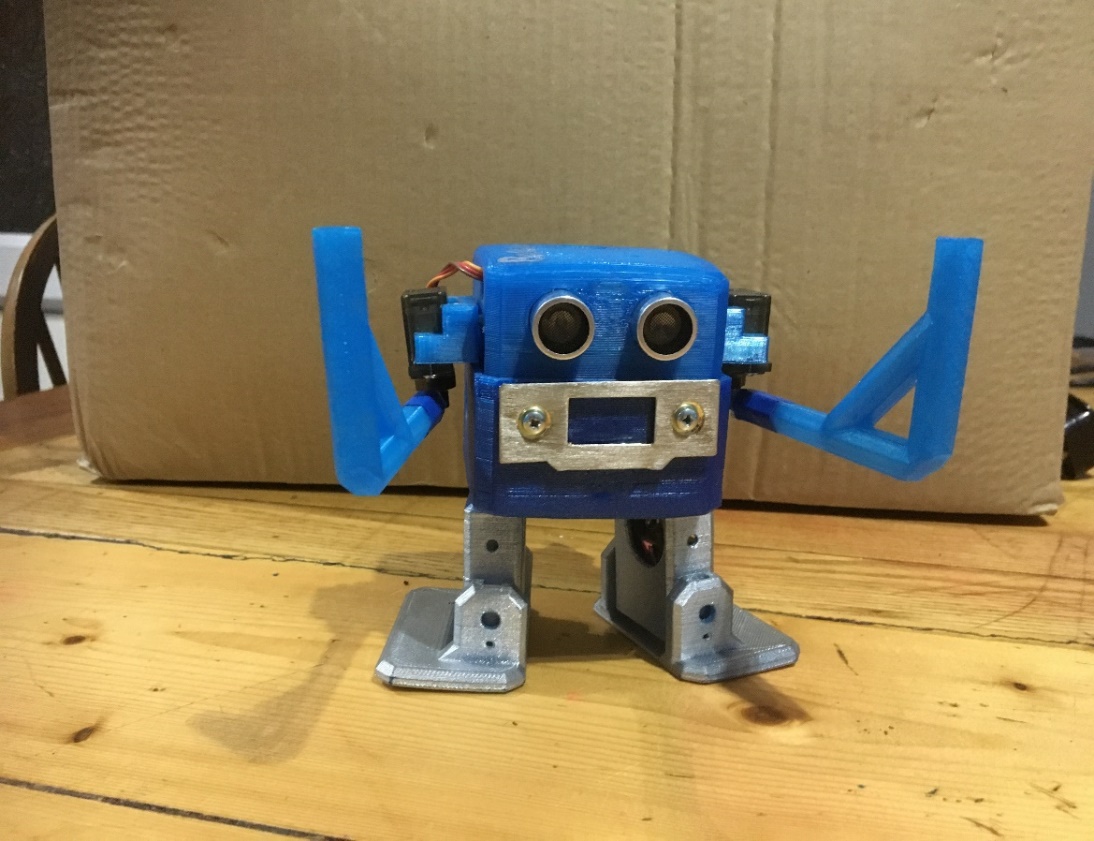


Fig 11. updated internal components Fig 12. Updated assembled device

**Documented design, software:**

The software implementation was processed in partitions, beginning with the conjunctive use of the ultrasonic sensor and servo motors.

I wrote a test program which relayed relevant results to cause servo motor movement, this allowed for testing of both the hardware components and the ability to translate one medium of data to another.

#include <Servo.h>

const int trig = 9;

const int echo = 8;

long duration;

int distance;

void setup() {

pinMode(trig, OUTPUT);

pinMode(echo, INPUT);

Serial.begin(9600);

}

void loop() {

digitalWrite(trig, LOW);

delay (2);

digitalWrite(trig, HIGH);

delay (10);

digitalWrite(trig, LOW);

duration = pulseIn(echo, HIGH);

distance= duration\*0.034/2;

Serial.print(distance);

}

Using the information received from this code, the data can then be used as seen in further code examples.

Once validation of hardware was completed, focus was based on the movement of the device, the initial walk cycle was very heavily based on the use of the servo collect set function.

Along with other Arduino functions, such as exit (0) and serial functions, the remainder of the devices function is a composition of all previously mentioned.

Additions vary on walk functions to account for left and right steps from the initial position and left and right steps once the legs are situated with a 30 degrees variance from moving forward.

**Technical solution:**

Through the technical solution I have achieved a large percentage of the given objectives.

Out of the given objectives below, I have been able to achieve all of the objectives with exception of the ease of access to the device, as a result of its complexity, however this is still possible but arguably, not in a user friendly manner.

Secondly, I have only been able to partially achieve the objective to make the user aware of the device being in an uncertifiable condition.

* The device should be capable of processing information and locating the suitable data in order to confirm its next function.
* It should be able to operate completely independently without the use of a computer except from a user recording data feedback.
* The device should be able to be easily altered and have accessible maintenance to components such as sensors and the microcontroller board.
* The device should be capable of alerting the user to a situation which is not rectifiable.
* It should include a fail-safe function for when no further actions can be taken.
* The device should have an operational independent lifetime of up to ten minutes as a minimum.
* The device should be able to output error messages, data inputs and any necessary information to the user when connected to the Arduino ide serial monitor.
* The device should be able to recognize situations where no further action can be taken and halt operation.
* The device should have appropriate conditions to verify the condition of the device.
* The device should have a suitable durability to deal with a variety of scenarios which it may encounter.
* The device should be full contained as to prevent issues with regards to the hardware's integrity.

Overall, all objectives have been attempted and most of the objectives have been achieved.

Below is the completed solution for the investigation.

/\* N.E.A Boswin 3.0 project

\*

\* servo notation diagram.

\*

\* for pin locators use servo. Attach functions.

\*

\* L | R

\* --------------|-------------

---------------

\_\_ | O O | \_\_

\ \\_\_\_\_|---------------|\_\_\_\_/ /

\-----| |-----/ =s5(left arm) / s6(right arm)

---------------

|| || =s3(left leg) / s4(right leg)

|| ||

----- ------

|----- ------| =s1(left foot) / s2(right foot)

focus based around the devices receptiveness of ultrasonic frequencies are allocated within the main loop.

\*/

//libraries

#include <Servo.h>

//initializers and variables

#define off\_button 8

//assorted servo notation from diagram

Servo s1;

Servo s2;

Servo s3;

Servo s4;

Servo s5;

Servo s6;

int pressed;// check state of button

int pos = 90;// constant calibration value

const int trig = 10;

const int echo = 9;

long duration;

int distance;

int s1\_set;

int s2\_set;

int s3\_set;

int s4\_set;

int s5\_set;

int s6\_set;

//servo setting function, sets all servos using a single function.

void servo\_collect\_set(int s1\_set, int s2\_set, int s3\_set, int s4\_set, int s5\_set, int s6\_set){

s1.write(s1\_set);

s2.write(s2\_set);

s3.write(s3\_set);

s4.write(s4\_set);

s5.write(s5\_set);

s6.write(s6\_set);

}

void setup() {

//servo connections established

//notation in brackets refer to physical connection on arduino Nano.

s1.attach(2);

s2.attach(3);

s3.attach(4);

s4.attach(5);

s5.attach(6);

s6.attach(7);

//initialise fail safe button.

pinMode(off\_button, INPUT);

digitalWrite(off\_button, HIGH); // connect internal pull-up resistor.

//create ultrasonic sensor connections to arduino.

pinMode(trig, OUTPUT);

pinMode(echo, INPUT);

//start serial monitor for feedback results.

Serial.begin(9600);

}

void loop() {

servo\_collect\_set(pos, pos, pos, pos, pos, pos);

//set arm values to stationary position.

delay(500);

s5.write(pos-90);

s6.write(pos+90);

delay(200);

//sets ultrasonic sensor to start state.

digitalWrite(trig, LOW);

delay (2);

digitalWrite(trig, HIGH);

delay (10);

digitalWrite(trig, LOW);

duration = pulseIn(echo, HIGH);

distance= duration\*0.0343/2;//distance using speed of sound constant, can be varied for accuracy.

if (distance != 0){

Serial.print(distance);

Serial.print("cm ,");

if (distance<25){

servo\_collect\_set(pos, pos, pos, pos, pos, pos);

delay(500);

s5.write(0);

s6.write(180);

delay(500);

}

else if (distance>25){

servo\_collect\_set(pos, pos, pos, pos, pos-90, 180);

servo\_collect\_set(pos-85, pos, pos, pos, pos-90, 180);

delay(50);

servo\_collect\_set(pos-85, pos-40, pos, pos, pos-90, 180);

delay(625);

servo\_collect\_set(pos-30, pos-40, pos, pos, pos-90, 180);

delay(1250);

servo\_collect\_set(pos-30, pos-40, pos+10, pos+10, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+20, pos+20, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+30, pos+30, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+30, pos+30, pos, pos);

delay(625);

servo\_collect\_set(pos, pos, pos+30, pos+30, pos-90, 180);

delay(1000);

// step 1

servo\_collect\_set(pos, pos+85, pos+30, pos+30, pos-90, 180);

delay(50);

servo\_collect\_set(pos+40, pos+85, pos+30, pos+30, pos, pos);

delay(625);

servo\_collect\_set(pos+40, pos+30, pos+30, pos+30, pos, pos);

delay(1250);

servo\_collect\_set(pos+40, pos+30, pos+30, pos+30, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos+20, pos+20, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos+10, pos+10, pos, pos);

delay(50);

servo\_collect\_set(pos, pos, pos, pos, pos-90, 180);

delay(1000);

//step 2

servo\_collect\_set(pos, pos+85, pos, pos, pos-90, 180);

delay(50);

servo\_collect\_set(pos+40, pos+85, pos, pos, pos, pos);

delay(625);

servo\_collect\_set(pos+40, pos+30, pos, pos, pos, pos);

delay(1250);

servo\_collect\_set(pos+40, pos+30, pos-10, pos-10, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos-20, pos-20, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos-30, pos-30, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos-30, pos-30, pos, pos);

delay(625);

servo\_collect\_set(pos, pos, pos-30, pos-30, pos-90, 180);

delay(1000);

//step 3

servo\_collect\_set(pos-85, pos, pos-30, pos-30, pos-90, 180);

delay(50);

servo\_collect\_set(pos-85, pos-40, pos-30, pos-30, pos, pos);

delay(625);

servo\_collect\_set(pos-30, pos-40, pos-30, pos-30, pos, pos);

delay(1250);

servo\_collect\_set(pos-30, pos-40, pos-20, pos-20, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos-10, pos-10, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos, pos, pos, pos);

delay(50);

servo\_collect\_set(pos, pos, pos, pos, pos-90, 180);

delay(1000);\

//step 4

servo\_collect\_set(pos, pos, pos, pos, pos-90, 180);

servo\_collect\_set(pos-85, pos, pos, pos, pos-90, 180);

delay(50);

servo\_collect\_set(pos-85, pos-40, pos, pos, pos-90, 180);

delay(625);

servo\_collect\_set(pos-30, pos-40, pos, pos, pos-90, 180);

delay(1250);

servo\_collect\_set(pos-30, pos-40, pos+10, pos+10, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+20, pos+20, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+30, pos+30, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+30, pos+30, pos, pos);

delay(625);

servo\_collect\_set(pos, pos, pos+30, pos+30, pos-90, 180);

delay(1000);

//step 5

servo\_collect\_set(pos, pos+85, pos+30, pos+30, pos-90, 180);

delay(50);

servo\_collect\_set(pos+40, pos+85, pos+30, pos+30, pos, pos);

delay(625);

servo\_collect\_set(pos+40, pos+30, pos+30, pos+30, pos, pos);

delay(1250);

servo\_collect\_set(pos+40, pos+30, pos+30, pos+30, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos+20, pos+20, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos+10, pos+10, pos, pos);

delay(50);

servo\_collect\_set(pos, pos, pos, pos, pos-90, 180);

delay(1000);

//step 6

servo\_collect\_set(pos, pos, pos, pos, pos-90, 180);

servo\_collect\_set(pos-85, pos, pos, pos, pos-90, 180);

delay(50);

servo\_collect\_set(pos-85, pos-40, pos, pos, pos-90, 180);

delay(625);

servo\_collect\_set(pos-30, pos-40, pos, pos, pos-90, 180);

delay(1250);

servo\_collect\_set(pos-30, pos-40, pos+10, pos+10, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+20, pos+20, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+30, pos+30, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+30, pos+30, pos, pos);

delay(625);

servo\_collect\_set(pos, pos, pos+30, pos+30, pos-90, 180);

delay(1000);

//step 7

servo\_collect\_set(pos, pos+85, pos+30, pos+30, pos-90, 180);

delay(50);

servo\_collect\_set(pos+40, pos+85, pos+30, pos+30, pos, pos);

delay(625);

servo\_collect\_set(pos+40, pos+30, pos+30, pos+30, pos, pos);

delay(1250);

servo\_collect\_set(pos+40, pos+30, pos+30, pos+30, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos+20, pos+20, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos+10, pos+10, pos, pos);

delay(50);

servo\_collect\_set(pos, pos, pos, pos, pos-90, 180);

delay(1000);

}

}

delay(500);

digitalWrite(trig, LOW);

delay (50);

digitalWrite(trig, HIGH);

delay (50);

digitalWrite(trig, LOW);

//button check

pressed = digitalRead(off\_button);

if (pressed==LOW)

{

Serial.write("end error 1");

delay(2000);

exit(0);

}

if (distance<25)// if area is not clear rotate right approximately 90 degrees.

{

servo\_collect\_set(pos, pos, pos, pos, pos-90, 180);

servo\_collect\_set(pos-85, pos, pos, pos, pos-90, 180);

delay(50);

servo\_collect\_set(pos-85, pos-40, pos, pos, pos-90, 180);

delay(625);

servo\_collect\_set(pos-30, pos-40, pos, pos, pos-90, 180);

delay(1250);

servo\_collect\_set(pos-30, pos-40, pos+10, pos+10, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+20, pos+20, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+30, pos+30, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+30, pos+30, pos, pos);

delay(625);

servo\_collect\_set(pos, pos, pos+30, pos+30, pos-90, 180);

delay(1000);

//step 1

servo\_collect\_set(pos, pos+85, pos+30, pos+30, pos-90, 180);

delay(50);

servo\_collect\_set(pos+40, pos+85, pos+30, pos+30, pos, pos);

delay(625);

servo\_collect\_set(pos+40, pos+30, pos+30, pos+30, pos, pos);

delay(1250);

servo\_collect\_set(pos+40, pos+30, pos+30, pos+30, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos+20, pos+20, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos+10, pos+10, pos, pos);

delay(50);

servo\_collect\_set(pos, pos, pos, pos, pos-90, 180);

delay(1000);

//step 2

servo\_collect\_set(pos, pos, pos, pos, pos-90, 180);

servo\_collect\_set(pos-85, pos, pos, pos, pos-90, 180);

delay(50);

servo\_collect\_set(pos-85, pos-40, pos, pos, pos-90, 180);

delay(625);

servo\_collect\_set(pos-30, pos-40, pos, pos, pos-90, 180);

delay(1250);

servo\_collect\_set(pos-30, pos-40, pos+10, pos+10, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+20, pos+20, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+30, pos+30, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+30, pos+30, pos, pos);

delay(625);

servo\_collect\_set(pos, pos, pos+30, pos+30, pos-90, 180);

delay(1000);

//step 3

servo\_collect\_set(pos, pos+85, pos+30, pos+30, pos-90, 180);

delay(50);

servo\_collect\_set(pos+40, pos+85, pos+30, pos+30, pos, pos);

delay(625);

servo\_collect\_set(pos+40, pos+30, pos+30, pos+30, pos, pos);

delay(1250);

servo\_collect\_set(pos+40, pos+30, pos+30, pos+30, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos+20, pos+20, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos+10, pos+10, pos, pos);

delay(50);

servo\_collect\_set(pos, pos, pos, pos, pos-90, 180);

delay(1000);

if (distance<25)// if distance 90 degrees to right not clear turn 90 degrees further.

{

//step 1

servo\_collect\_set(pos, pos, pos, pos, pos-90, 180);

servo\_collect\_set(pos-85, pos, pos, pos, pos-90, 180);

delay(50);

servo\_collect\_set(pos-85, pos-40, pos, pos, pos-90, 180);

delay(625);

servo\_collect\_set(pos-30, pos-40, pos, pos, pos-90, 180);

delay(1250);

servo\_collect\_set(pos-30, pos-40, pos+10, pos+10, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+20, pos+20, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+30, pos+30, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+30, pos+30, pos, pos);

delay(625);

servo\_collect\_set(pos, pos, pos+30, pos+30, pos-90, 180);

delay(1000);

//step 2

servo\_collect\_set(pos, pos+85, pos+30, pos+30, pos-90, 180);

delay(50);

servo\_collect\_set(pos+40, pos+85, pos+30, pos+30, pos, pos);

delay(625);

servo\_collect\_set(pos+40, pos+30, pos+30, pos+30, pos, pos);

delay(1250);

servo\_collect\_set(pos+40, pos+30, pos+30, pos+30, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos+20, pos+20, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos+10, pos+10, pos, pos);

delay(50);

servo\_collect\_set(pos, pos, pos, pos, pos-90, 180);

delay(1000);

//step 3

servo\_collect\_set(pos, pos, pos, pos, pos-90, 180);

servo\_collect\_set(pos-85, pos, pos, pos, pos-90, 180);

delay(50);

servo\_collect\_set(pos-85, pos-40, pos, pos, pos-90, 180);

delay(625);

servo\_collect\_set(pos-30, pos-40, pos, pos, pos-90, 180);

delay(1250);

servo\_collect\_set(pos-30, pos-40, pos+10, pos+10, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+20, pos+20, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+30, pos+30, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+30, pos+30, pos, pos);

delay(625);

servo\_collect\_set(pos, pos, pos+30, pos+30, pos-90, 180);

delay(1000);

//step 4

servo\_collect\_set(pos, pos+85, pos+30, pos+30, pos-90, 180);

delay(50);

servo\_collect\_set(pos+40, pos+85, pos+30, pos+30, pos, pos);

delay(625);

servo\_collect\_set(pos+40, pos+30, pos+30, pos+30, pos, pos);

delay(1250);

servo\_collect\_set(pos+40, pos+30, pos+30, pos+30, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos+20, pos+20, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos+10, pos+10, pos, pos);

delay(50);

servo\_collect\_set(pos, pos, pos, pos, pos-90, 180);

delay(1000);

if (distance<25)//if distance 180 degrees from original direction turn 90 degrees.

{

//step 1

servo\_collect\_set(pos, pos, pos, pos, pos-90, 180);

servo\_collect\_set(pos-85, pos, pos, pos, pos-90, 180);

delay(50);

servo\_collect\_set(pos-85, pos-40, pos, pos, pos-90, 180);

delay(625);

servo\_collect\_set(pos-30, pos-40, pos, pos, pos-90, 180);

delay(1250);

servo\_collect\_set(pos-30, pos-40, pos+10, pos+10, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+20, pos+20, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+30, pos+30, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+30, pos+30, pos, pos);

delay(625);

servo\_collect\_set(pos, pos, pos+30, pos+30, pos-90, 180);

delay(1000);

//step 2

servo\_collect\_set(pos, pos+85, pos+30, pos+30, pos-90, 180);

delay(50);

servo\_collect\_set(pos+40, pos+85, pos+30, pos+30, pos, pos);

delay(625);

servo\_collect\_set(pos+40, pos+30, pos+30, pos+30, pos, pos);

delay(1250);

servo\_collect\_set(pos+40, pos+30, pos+30, pos+30, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos+20, pos+20, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos+10, pos+10, pos, pos);

delay(50);

servo\_collect\_set(pos, pos, pos, pos, pos-90, 180);

delay(1000);

//step 3

servo\_collect\_set(pos, pos, pos, pos, pos-90, 180);

servo\_collect\_set(pos-85, pos, pos, pos, pos-90, 180);

delay(50);

servo\_collect\_set(pos-85, pos-40, pos, pos, pos-90, 180);

delay(625);

servo\_collect\_set(pos-30, pos-40, pos, pos, pos-90, 180);

delay(1250);

servo\_collect\_set(pos-30, pos-40, pos+10, pos+10, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+20, pos+20, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+30, pos+30, pos, pos);

delay(50);

servo\_collect\_set(pos-30, pos-40, pos+30, pos+30, pos, pos);

delay(625);

servo\_collect\_set(pos, pos, pos+30, pos+30, pos-90, 180);

delay(1000);

//step 4

servo\_collect\_set(pos, pos+85, pos+30, pos+30, pos-90, 180);

delay(50);

servo\_collect\_set(pos+40, pos+85, pos+30, pos+30, pos, pos);

delay(625);

servo\_collect\_set(pos+40, pos+30, pos+30, pos+30, pos, pos);

delay(1250);

servo\_collect\_set(pos+40, pos+30, pos+30, pos+30, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos+20, pos+20, pos, pos);

delay(50);

servo\_collect\_set(pos+40, pos+30, pos+10, pos+10, pos, pos);

delay(50);

servo\_collect\_set(pos, pos, pos, pos, pos-90, pos+90);

delay(1000);

if (distance<25)// if distance is completely obstructed end procedure.

{

servo\_collect\_set(pos, pos, pos, pos, pos-90, pos+90);

delay(2000);

servo\_collect\_set(pos, pos, pos, pos, pos, pos);

delay(2000);

servo\_collect\_set(pos, pos, pos, pos, pos+90, pos-90);

delay(2000);

Serial.write("end error 2");

delay(2000);

exit(0);

}

}

}

}

}

The devices code operates based around feedback generated from the ultrasonic sensor, using the information to negotiate the next most viable function.

The device has two possible failsafe options, these two options only occur after one of two possible conditions have been fulfilled, the first of which being if the device has verified that an obstacle has intercepted a 25 cm range, three consecutive times meaning that there is no feasible further actions, therefore signifying operation has ended by the arms being rotated fully backwards and no further movement.

The second of the two failsafe's is triggered when the device has fallen onto a micro switch located on the back of the device, this when triggered will halt all actions and also complete the previously mentioned arm motion, in an attempt to prevent any damage to the device.

As previously mentioned, there is the inclusion of the servo collect set function which allows values to be altered easily by being able to recognize individual actions.

Furthermore, the code provides essential readouts to the Arduino serial monitor when connected to a computer.

By having access to real time information, the device can be debugged easily and removes a high degree of ambiguity from the devices testing.

In conclusion, the devices source code is designed around potential errors and the utilization of valuable information, allowing the device’s operation to be as successful as possible to prevent issues When using the device to confirm the ultrasonic sensors effectiveness for being used for guidance of the device.

## Testing phase:

As a result of the device being a robotics project testing is only possible through a physical medium.

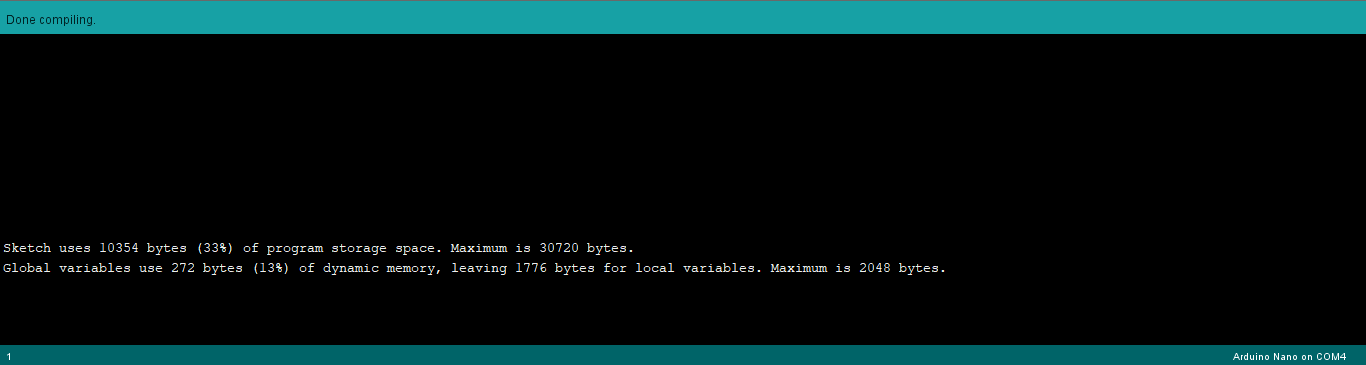
In terms of the code's operation within the Arduino IDE, compilation the code is successful using approximately 33% of the Arduino Nano’s internal storage as well as 13% of the devices dynamic memory as shown in figure 13.

Figure .13 compilation verification.

Once uploaded to the device, all further evidence will be provided through video.

Testing will be done to analyse the device’s:

* Walk cycle
* Sensing ability

All will be provided through thingiverse at :<https://www.thingiverse.com/make:767641#comment-2926273>

In conclusion from testing the formatting of the devices logical testing worked as expected as can be seen when the device approaches a barrier it tests 3 times to see if the obstacle is present however since the device was unable to proceed it incorporated the exit function and provided an indicator that the device is no longer operating, this was signified by the two arms rotating behind the central body of the device. From this it allows the user to recognise unescapable errors. Furthermore, the test displayed that from the point it was initially sensing an obstacle, the device was able to take an appropriate course of action by not colliding with the obstacle at all.

From this test, the devices sensing ability was verified to a suitable accuracy however further testing would be needed to verify the true accuracy of the device range finding.

Further developments were shown in the evidence of the walk cycle, this demonstrated how the device was able to utilise its functions to stabilise and control its movement.

The walk cycle is integral to the device's operation, therefore is a significant success to the device's operation

From tests, the relayed information to the Arduino ide is highly accurate calculating values to a suitable degree with 0.5 cm deviations, as a result the sensor worked very well.

However, when working with solid materials, the device had extremely successful recognition abilities, even sensing distances up to 20 metres, unfortunately though the device has limitations of the sensors function, as a result of the devices focus on echo locative navigation, issues have been caused when tested with extreme low density materials.

As a result of poor or no returned values when detecting low density materials, the device is not fully reliable for detecting all material.

To improve this factor the device would require the inclusion of further ultrasonic sensors to make approximations of the real-world inputs, using an average generated from several sensors would help give a more reliable approximation for use in future experiments.

As a result of tests, the devices operation was altered to limit the possibility of failure, such as the use of the arms and the addition of failsafe functions which prevent the device continuing operation which puts the device at risk.

From the developments the device has been highly successful in standard operation.

Overall, from testing the device showed a high degree of receptiveness of its environment and ability to deal with difficulties.

Areas of weakness are intended to be improved upon in further project developments, however, is not necessary, as when tested in natural environments, the devices sensitivity was highly successful and is viable for the investigation.

## Evaluation:

In conclusion the device is to high degree capable of recognizing obstacles and managing the usage of data received to conduct its next movements.

The device, with regards to its hardware is highly effective and has very minimal areas of deficit, those areas being is the uncertainty with regards to sensing low density materials, to overcome this the addition of further ultrasonic possibly using the approximate values in conjunction to develop the accuracy of the results.

From the device's investigations, it has been discovered that using ultrasonic bulk wave locating is highly viable for the most part.

The device was able to recognize a range of dense materials to deal with typical material compositions, however two main exceptions were found.

The first of the two errors were the obscuring of reception via particle interference.

To elaborate, the device is prone to a potential error when the sensors are placed in an environment which would cause deflection to the ultrasonic rays.

This is theorized to be caused as the initial wavelengths are not retuned giving unverified response to the sensor.

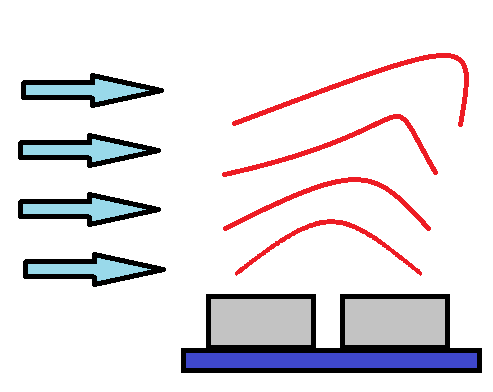
Figure 14 represents the potential issue.

Fig 14, the effect of external forces on the regularity of wave lengths.

As seen in the diagram, the red irregular lines are representative of the output from the ultrasonic sensor, and the blue arrows signify an external force, most typically excessive air flow/ winds.

As seen the blue arrows cause interference with the ultrasonic waves, as a result the device is incapable of having the output returned as a result to the skewing of the wavelengths.

As previously mentioned, a second issue occurred with regards to the ultrasonic detection based on low density material.

The issue is as a result of the ultrasonic waves passing through the material due to a lack of reflective materials.

As seen in figure 15 the porous nature of the material causes potential issues

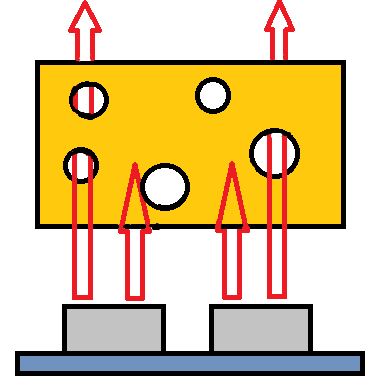


Fig 16 represents how the waves given from the ultrasonic sensor Is prone to being misdirected as a result of porous properties, as a result the returned values are either reduced in accuracy or not available.

In conclusion, given the two mentioned areas where the device has issues with the engineering with regards to the ultrasonic sensor function, the use is highly effective in comparison to other mediums of detection and verification.

Due to the lack of dependency on visual input, like that a visual based display, the feedback received is purely based on physical attributes rather than approximations of the environment.

As a result of set results being given the security of a value being correct is much more likely than algorithms using cameras for predictive values.

So, to elaborate, the use of an ultrasonic sensor is highly suitable for range finding, however given the areas of deficiency, the use would be most beneficial when used in conjunction with another sensing device to verify and coordinate results between the two inputs.

Onn the other hand as a single, low powered a mostly reliable sensor the application is highly versatile and potential applications from this investigation allows for the limitations to be accounted for and expanded upon in future projects.