**Embedded Systems Project Technical Report**

Christopher Gidney - cg930

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

**Sections:**

Pg. 2 - **R1 Project Summary**

Pg. 4 - **R3 Technical Description of the Group Component Implementation**

Pg. 7 - **R4 Evidence of Testing of Group Implementation**

Pg. 10 - **R5 Technical Description of Individual Component Implementation**

Pg. 14 - **R7 Evidence of Preparation for Assessment and Team Management**

**R1 Project Summary**

**Group Project Goals**

The group project goals were to implement a working solution using the MBED board, servo and connected infrared and ultrasonic sensors for each of the following points:

* A ‘tape measure mode’ that measures distance and outputs a constantly updated value to LCD display
* A ‘calibration mode’ that can be used to calibrate the distance measurements of both the infrared and ultrasonic distance sensors, with a user interface to guide a user through calibration
* A ‘scan mode’ that rotates the servo taking distance measurements at at least 20 points through its rotation
* A ‘multi view mode’ that can combine multiple scan sweeps of an object at different rotations to create an all-round profile of the object
* A user interface combining a keypad and LCD to allow relevant parameters to be modified and to display relevant information that ties in with the above four modes.
* Raw data that is displayed on an external PC
* Graphical representation of scan data and mode, with real time updates

Each of these goal were to be completed to a satisfactorily working standard or better, with equal and fair contribution from each of the four group members.

**Group Project Outcomes**

Most of the group project goals were met to a working standard. Tape measure mode was implemented properly and worked very well. Calibration mode was implemented to a working stage, however with quite basic calibration methods for each of the sensors. Scan mode was implemented very well allowing a good number of parameters to be set for each scan. Multi view mode however was not implemented. Reasons for this are as a group we were unsure of how this would be implemented and so it was left to the end so we could focus on the other tasks first. Also certain tasks took longer than expected due to being less straightforward than originally thought and so took up time that would have otherwise been used to work on multi view mode.

A good user interface was implemented that used both the LCD and keypad and does a good job of tying all of the modes together. Raw data can be displayed on an external PC but only with limited data being displayed. And finally a graphical representation of the scan mode data was implemented, however it was not the best solution and technically did not update in real time.

As a group we felt we worked well together, all contributing a fair amount to the project.

**Individual Project Goals**

The individual project goals were to implement a working object tracker using the MBED board, servo and connected infrared and ultrasonic sensors. This solution should interface with existing group solutions, building on them to create the tracker. The object tracker should follow an object that is placed in front of the sensors by sweeping the servo back and forth detecting each edge of the object.

**Individual Project Outcomes**

The individual project goals were met with a working object tracker being created. The tracker would follow a decently sized object well, sweeping across the object from left to right with each edge being detected. It had some flaws as small objects or objects far away could not be reliably tracked and good sized objects that were moved too fast could also not be reliably tracked.

The solution could have been improved by using both sensors instead of just one and by perfecting the main control loop which determines how the servo should move.

**R3 Technical Description of Group Component Implementation**

**Infrared Distance Sensor and Infrared Sensor Calibration**

The infrared distance sensor used is a Sharp GP2Y0A21YK0F which can measure accurately distances from 10 to 80cm. The sensor outputs an analogue signal and so is connected to the analogue-to-digital converter input on pin number 19 of the MBED board. The ADC result is accessed from ADC channel 4, which will be a value from 0 to 4095. From testing the IR sensor at multiple distances and plotting ADC result over distance on a graph, a mathematical function for working out distance based on ADC result was calculated. This function is as follows:

distance(cm) = (multiplier x adcValue)-0.979

The multiplier is set to 26530 by default before calibration and is changed based on sensor calibration results. When an IR sensor distance value is requested in code, 10 distance values are calculated and the median value of these is outputted. This is to provide a more accurate output by preventing erroneous results from being included due to signal noise for example.

The IR sensor is calibrated by inputting two distance measurement values, one at 10cm and one at 50cm. The multiplier that would individually fit each of the two measurements the best is calculated by rearranging the earlier formula like so:

multiplier = adcValue / distance(cm)-0.979

The average of these two multiplier values is then calculated and this value replaces the default multiplier. Obviously this is a very basic method of calibration as only the multiplier value is modified and only two distances are used for calibration. If more distance values were used and the exponent -0.979 was modified, the IR sensor calibration could have been much more accurate.

**Ultrasound Sensor**

The ultrasound sensor used is a HC-SR04 that can measure accurately distances from 2 to 400cm. The sensor has one input and one output and these are both connected GPIO pins on the MBED board. The input *Trigger* or *Trig* is used to initialise the sensor so a reading can be output, to initialise it needs a 5V pulse of at least 10us. The output *Echo* will send a 5V pulse of which length is proportional to the distance.

The first part of operation of the sensor is to set the GPIO pin Trig is connected to high for a minimum of 10us. Once triggered the Echo output will be set to high (5V), and when this happens the microcontroller will record the time at which the corresponding GPIO pin signal rose. The Echo output will stay at 5V until the signal the ultrasound sensor returns, at which point the signal will drop to low (0V). When the signal drops to low the capture timer will record this time and the pulse length is calculated by subtracting one time from the other. The pulse length can then be used to calculate the distance between the ultrasound sensor and whatever is in front of it using this equation:

distance to object = speed of sound x pulse length / 2

The pulse length must be divided by 2 as the length of the pulse is the time the ultrasonic signal takes to get to the object and back.

**Servo**

Pulse width modulation is used to control the servo position. By sending a pulse of set length every 20 milliseconds to keep up with the servos 50 Hz polling rate the position can be set. Using PWM uses minimal processor time as once it is set it will continue to send a pulse of the set length until the pulse length is required to be changed or the PWM module is stopped. Using the PWM the servo can be set to one of eighty positions, where position 0 is the furthest left position and position 80 is the furthest right position.

**Scan Mode**

Scan mode uses the infrared sensor and servo to measure distances over a sweep of the servo. A minimum of twenty distance measurements are taken in a sweep and the number of measurements taken can be changed by modifying the samples per sweep parameter.

The following parameters can be modified to determine the scan: start position, stop position, samples per sweep and sweep speed

When the scan mode is started the servo moves to its set start position, takes a distance measurement, moves a set number of positions over, takes a distance measurement and repeats until its set stop position is reached. The number of positions that the servo moves between measurements is determined by the start position, stop position and samples per sweep. These three parameters are needed to ensure that the angle between each sample is the same. The number of positions to move between measurements is calculated in code like so:

positions to move = (stop position – start position) / samples

**LCD Display and User Interface**

The LCD displays a menu based user interface controlled by keypad allowing the user to access the different modes and change parameters. When the MBED is turned on the LCD displays briefly instructions on how to navigate the user interface. ‘\*’ and ‘#’ are used to cycle left and right between menu items, this is required as the LCD is limited to two lines of text at a time. ‘1’ is used to select a menu item and ‘0’ is used to ‘go back’ or exit from the previously selected menu item.

The menu structure is like so:

Functions

Calibration

Tape-Measure

Scan

Multi-View

Parameters

Sweep Speed

No. of Samples

Start Position

Stop Position

Choose Sensor

Infrared

Ultrasound

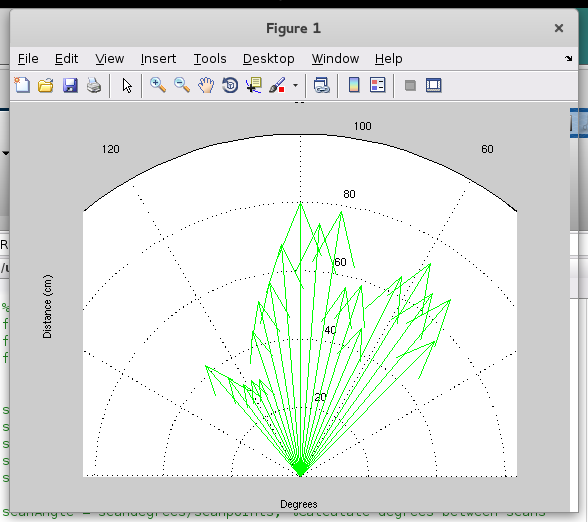
The user interface uses a set of nested loops checking for keypad inputs to allow the user to enter different parts of the menu. When a menu item is selected an inner nested loop is entered, and when the user returns from this menu item the inner loop is exited.

**R4 Evidence of Testing of Group Implementation**

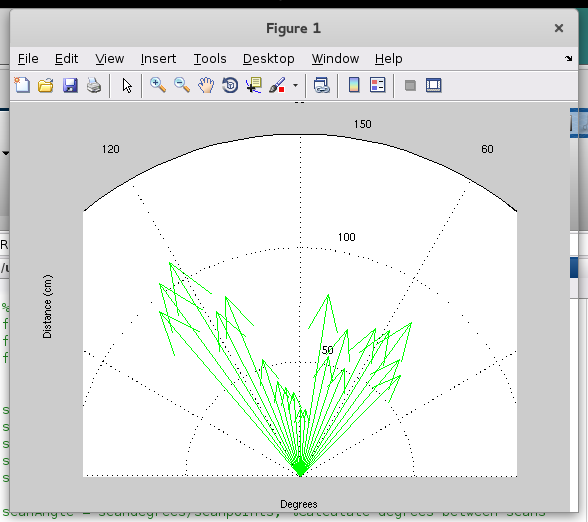
**Scan Mode Test & Results**

To test the scan mode and graphical display an object is placed in different areas within the scan sweep range and the graphical display is updated with the data from the scan.

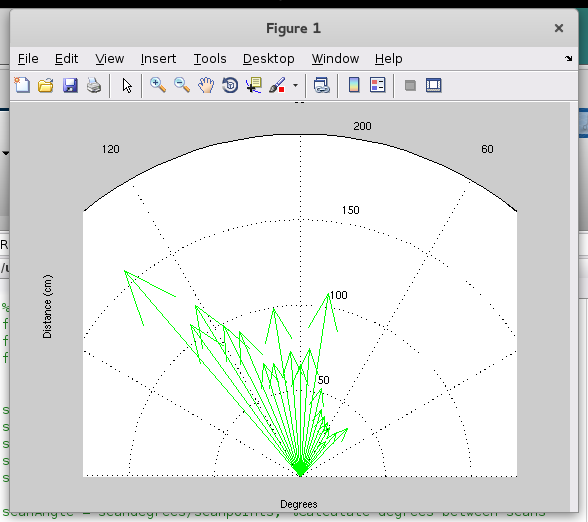
Object Placed at Left of Scan

****

Object Placed in Centre of Scan

****

Object Placed at Right of Scan

****

**Sensor Test & Results**

To test the infrared and ultrasound sensors at their default calibration settings, an object is placed at set distances in front of the sensor and the output from the code is compared to the expected output value.

For the infrared sensor at 5cm the expected outcome is any value which isn’t 5cm as the sensors operating range is 10 to 80cm and so at 5cm it is reasonable to expect the value outputted will not be 5cm.

|  |  |  |  |
| --- | --- | --- | --- |
| Object distance (cm) | Sensor | Expected outcome (cm) | Actual outcome (cm) |
| 5 | Infrared | Any value which isn’t 5 | 13.7 |
| 15 | Infrared | 15 | 14.6 |
| 25 | Infrared | 25 | 22.6 |
| 35 | Infrared | 35 | 31.2 |
|  |  |  |  |
| 5 | Ultrasound | 5 | 5.17 |
| 15 | Ultrasound | 15 | 14.69 |
| 25 | Ultrasound | 25 | 25.12 |
| 35 | Ultrasound | 35 | 34.71 |

**Calibration Test Results**

To test the calibration mode accuracy the infrared and ultrasound sensors are calibrated with correct distances and then with incorrect distances. The output from the code when an object is placed 20cm from the sensors is compared with the expected output in this situation to determine the accuracy of the calibration.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10cm Position (cm) | 50 cm Position (cm) | Sensor | Expected outcome at 20cm (cm) | Actual outcome at 20cm (cm) |
| 10 | 50 | Infrared | 20 | 22.1 |
| 10 | 50 | Ultrasound | 20 | 19.81 |
|  |  |  |  |  |
| 20 | 100 | Infrared | 10 | 11.9 |
| 5 | 25 | Infrared | 30 | 38.4 |
|  |  |  |  |  |
| 20 | 100 | Ultrasound | 10 | 19.78 |
| 5 | 25 | Ultrasound | 30 | 19.92 |

**R5 Technical Description of Individual Component Implementation**

**Overview – Object Tracking**

The individual component uses the infrared sensor and servo along with an edge detection algorithm to detect and follow an object which is placed in front of the sensor. The servo is used to move the IR sensor left and right across the object, detecting its left and right edges, and so will follow the object if it moves. The IR sensor was used in favour of the ultrasonic distance sensor as it has a narrower ‘cone’ in which distance measurements are taken which is more appropriate for edge detection, also the IR sensor is better suited for taking distance measurements on angled surfaces as some light will always be reflected back to the sensor.

The servo used can be set to one of eighty positions across a range of 90 degrees, so an array of size 80 is used to store the distance measurements at each position of the servo. This array of distance values is then used by the edge detection algorithm to detect changes in distance that should correspond to the edges of the object being tracked.

**Edge Detection Algorithm**

Figure 1, Edge Detection Stages

The edge detection algorithm uses a Sobel operator, in conjunction with thresholding and edge thinning to return an array containing positions of detected edges in the servo sweep.

The Sobel operator used is a 1 x 3 kernel like so [-1, 0, 1] that is convolved with the scan array to approximate the derivatives of the data in array. The Sobel operator is often used for edge detection in images, but can also be used in this case as the scan array is technically a 1 by 80 pixel image, where pixel intensity is a distance measurement. The output of the Sobel operator is an approximation of the gradient at each point in the scan array, so to find the edges in this output thresholding is used to find large enough changes in gradient to be edges. The thresholding part of the algorithm calculates the absolute difference between one element and the one after the next, i.e. the change in gradient over 3 elements, and if the difference is larger than a set threshold, a 1 is put in the corresponding place in the output edge array (0 representing no edge). From testing the edge detection algorithm as of so far it was obvious that there was a lot of cases where the edges in the array would have a thickness of more than 1 (e.g. [… 1, 1, …] is a thickness of 2). Edge thinning was implemented to reduce the thickness of each edge to a thickness of 1 to make it clearer where the edge of the object was found. The edge thinning part of the algorithm simply loops through the edge array to find edges of thickness greater than 1 and thins then to a thickness of 1. For example: [… 0, 1, 1, 1, 0 …] would become [… 0, 0, 1, 0, 0 …] after thinning. The final output of the edge detection algorithm is then a thinned edge array which is used in the main tracking control loop to determine the direction the servo should move next.

Below are two graphs showing the input and the output of the edge detection algorithm that was mocked up in MATLAB without edge thinning. In figure 2 the portion of the graph where the distance is lowest represents the object being tracked in the final solution. The two places where the distance jumps from around 20 to around 80, the edges are detected, represented as a 1 in the output array.

Figure 2, Input (sample) data

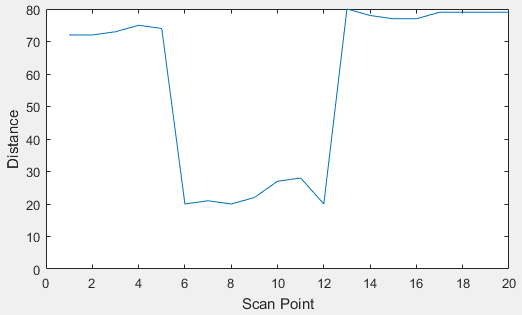
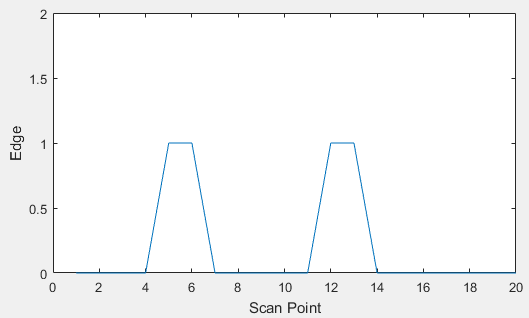
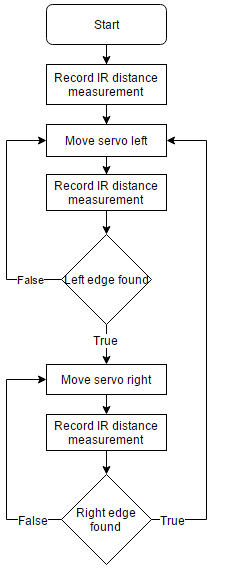


Figure 3, Output - Edges



**Main Tracking Control Loop**

Figure 4, Basic Tracking Control

The most basic premise of the main tracking control loop is: move left until an edge is found; move right until an edge is found; repeat. By following this the servo should sweep back and forth over the object and will follow the object if it moves.

The tracking control loop starts by setting the servo to its middle position and then waiting for a keypad ‘#’ press to start the tracking. The object to track must be placed in front of the infrared sensor when the tracking is started. When the tracking is started, the servo turns fully right and sweeps all the way to the left taking distance measurements at each of the eighty positions, each result being added to the scan array to fully populate it.

The servo then returns to the middle position and the tracking loop begins. Two variables determine the control flow in the loop: 1. edge – which is initially set to NONE 2. sweepdir – determines the direction of sweep and is initially set to LEFT.

Below is the rest of the control flow structure explained in pseudocode as it would be inconvenient to explain in prose.

Loop

**If edge is NONE**

**If sweep direction is LEFT**

Turn servo left

Wait 25ms

Update corresponding scan array position value to IR value

Run edge detection on scan array

**If edge is found at corresponding position**

Set edge to LEFT

**Else if servo is at edge of sweep range**

Set sweep direction to RIGHT

**Else if sweep direction is RIGHT**

Turn servo right

Wait 25ms

Update corresponding scan array position value to IR value

Run edge detection on scan array

**If edge is found at corresponding position**

Set edge to RIGHT

**Else if servo is at edge of sweep range**

Set sweep direction to LEFT

**Else if edge is LEFT and sweep direction is LEFT**

Set sweep direction to RIGHT

Set edge to NONE

Do twice

Turn servo right

Wait 25ms

Update corresponding scan array position value to IR value

**Else if edge is RIGHT and sweep direction is RIGHT**

Set sweep direction to LEFT

Set edge to NONE

Do twice

Turn servo left

Wait 25ms

Update corresponding scan array position value to IR value

Forever

Note there is a 25ms delay between turning the servo and taking an IR value reading to allow the servo to be turned properly before the reading is taken.

**Summary**

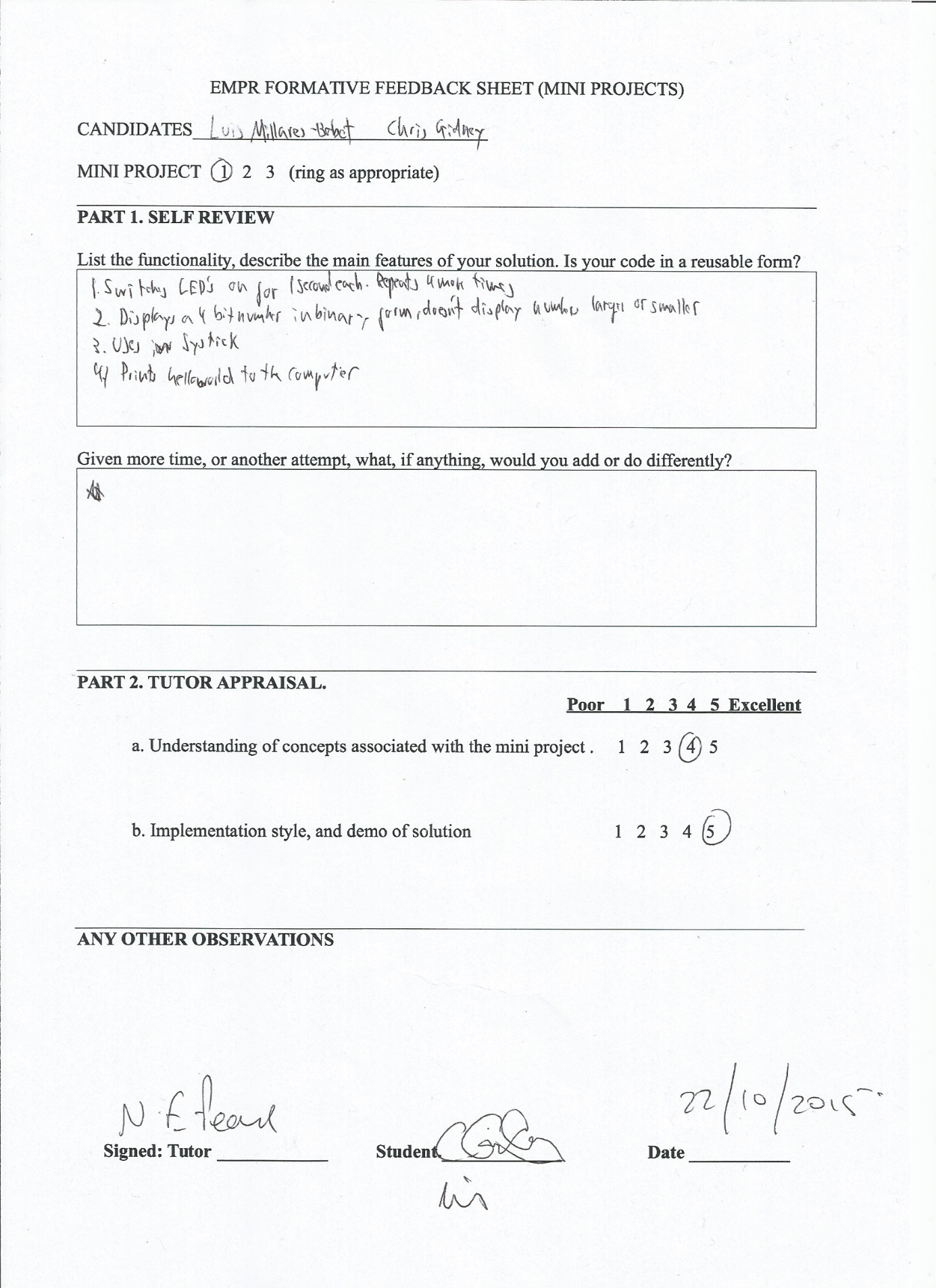
The object tracking works better in theory than it does in practice. Due to the very low resolution of the scan area, small objects or objects that are far away from the IR sensor will only take up a very small amount of the ‘pixels’ in the scan. This means that sweeping back and forth over these objects detecting each edge does not always work as the object could appear as only one edge in the scan, or the edges can be too close together which creates problems when sweeping.

There are two ways that come to mind in which the object tracking could be improved. The first is being able to set the servo to a greater number of positions than just eighty to improve the scan resolution and therefore have more accurate edge detection. The second way would be to use the IR sensor in conjunction with the ultrasound sensor to create a better ‘picture’ of the object in its environment so it can be tracked more accurately

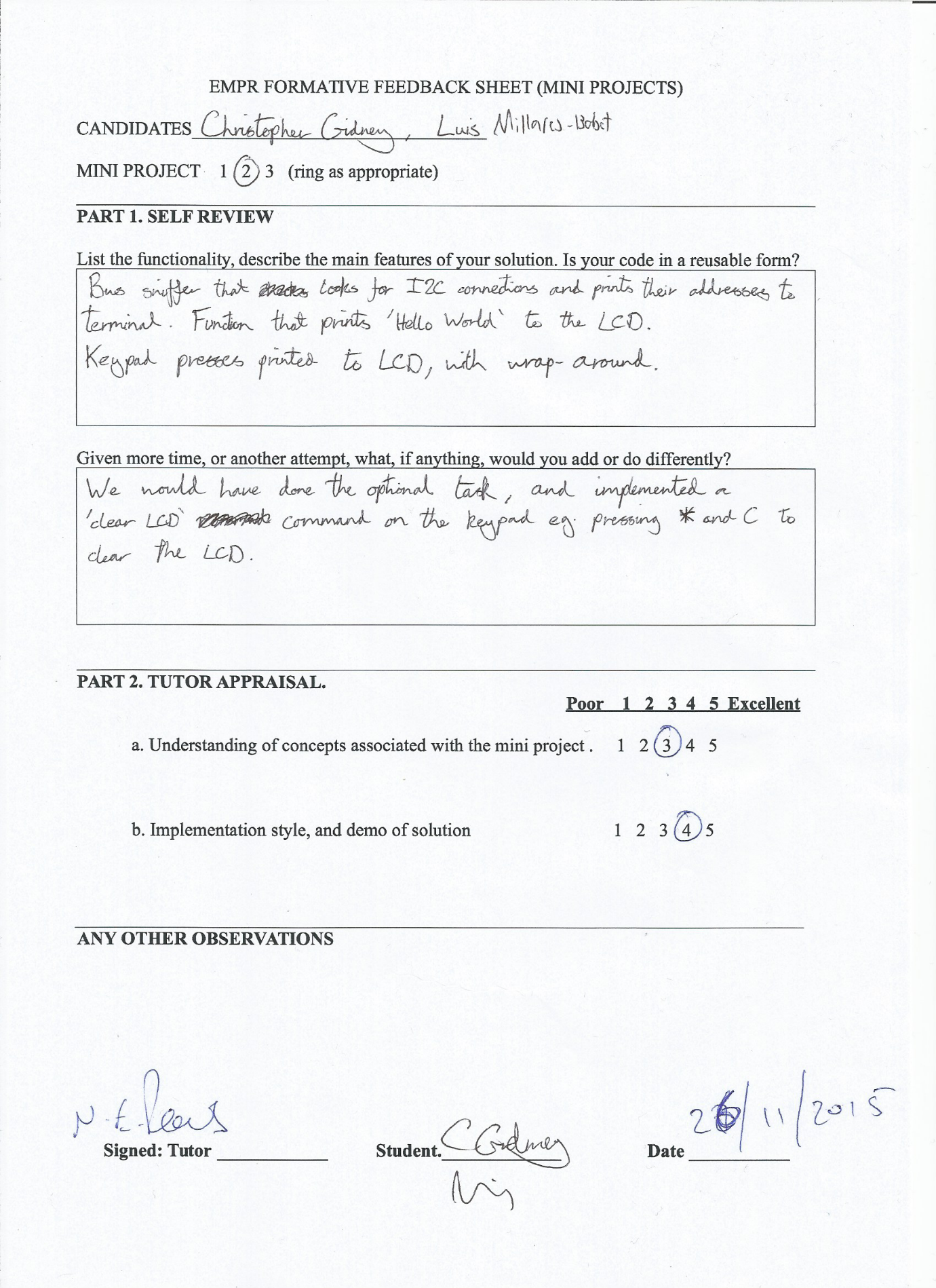
Overall the object tracking works okay for sizeable objects, and objects that are not moving too quickly.

**R7 Evidence of Preparation for assessment and team management**

**Mini Project 1 Feedback Sheet**

****

**Mini Project 2 Feedback Sheet**

****

**Group Minutes**

|  |  |  |
| --- | --- | --- |
| **Week** | **Set** | **Done** |
| 3 | Luis : UI( Hello, menu)  Chris : Infrared Sensor  Naz : Keypad Interrupts  Martin : PWM Interrupts | Luis : UI  Chris : Infrared Sensor  Naz : Fixed make file, fixed keypad interrupt code, timers  Martin : Made PWM, Got servos working |
| 4 | Luis : Calibration mode/ Tape Measure mode  Chris : Calibration mode /Tape Measure mode  Naz : Watchdog Timer for ultrasound  Martin : Ultrasound output translation |  |
| 5 | Luis : Calibration mode/ Tape Measure mode  Chris : Calibration mode/ Tape Measure mode  Naz : Other timers for ultrasound  Martin : Other timers for ultrasound | Naz : Ultrasound send signal done  Martin : Ultrasound send signal done |
| 6 | Luis : Calibration mode/ Tape Measure mode  Chris : Calibration mode/ Tape Measure mode  Naz : Other timers for ultrasound  Martin : Scan mode | Luis : Calibration mode/ Tape Measure mode and UI for each  Chris : Calibration mode/ Tape Measure mode and UI for each  Naz : GPIO and timer for ultrasound  Martin : Scan mode |
| 7 | Luis : Individual/ Ultrasound testing  Chris : Graphics  Naz : Individual / ultrasound testing  Martin : Sweep mode output to text document |  |