**Barcode**

|  |  |
| --- | --- |
| **Function** | **Use** |
| init\_IR\_barcode | Initialize the GP pins required for barcode (infrared) |
| code\_39\_decoder | Takes in the odd and even bits to decode the barcode, allocate the checksum and determine char based on predefined values, then return the char |
| determine\_odd\_bit | Takes in 5 odd bit timings to find the 2 highest and convert into ‘1’s while other bits will be ‘0’s |
| determine\_even\_bit | Takes in 4 even bit timings to find the highest and convert into ‘1’ while other bits will be ‘0’s |
| determine\_char | Gets one char at a time and keep track of the index of character being received. If the first char is the flipped delimiter, it will inform determine\_odd\_bit and determine\_even\_bit to flip the bits in the future conversion. |
| read\_barcode | Read the time taken for each bits within the 9 bits, always starting with black bit. On first black, will start count until it turns white and vice versa, until it reaches 9 bits.  Upon reaching 9 bits, 5 odd bits timings will be passed to determine\_odd\_bit and 4 even bits timings will be passed to determine\_even\_bit.  Thereafter, the converted bit will be passed to code\_39\_decoder which will decode the value and lastly, the decoded value will be passed to determine\_char. |

**Encoder**

|  |  |
| --- | --- |
| **Function** | **Use** |
| init\_encoder | Initialize the GP pins required for encoder |
| left\_encoder\_callback | Upon left wheel interrupt, add 1 to distance travelled and left wheel interrupt count |
| right\_encoder\_callback | Upon right wheel interrupt, add 1 to distance travelled and left wheel interrupt count |
| performance\_track\_task | Print the distance travelled and print the speed of left and right wheel every second and thereafter reset the left and right wheel interrupt count |

**Irline**

|  |  |
| --- | --- |
| **Function** | **Use** |
| init\_IR | Initialize the GP pins required for infrared |
| read\_IR | Read the digital value of infrared pin and return the integer |
| read\_task | Repeatedly calls read\_IR to get the reading |
| print\_task | Print the reading based on what was received from read\_task |

**Magnetometer**

|  |  |
| --- | --- |
| **Function** | **Use** |
| init\_GY\_511 | Initialize the GP pins required for magenetometer |
| change\_to\_continuous | Change the setting for magnetometer to continuously read data |
| request\_magneto\_read | Sends the memory address of the register to read from, get the next 6 bytes and convert into x, y, z value. Thereafter, convert the reading of x and y into degrees. |

**Maze Solving**

|  |  |
| --- | --- |
| **Function** | **Use** |
| mapInit | Create an array of the map, whilst initializing all it’s value |
| resetVisited | Reset the map visited value, marking all nodes as unvisited |
| resetDirection | Reset the map direction, marking all nodes as directionless |
| setWall | Upon detecting a wall, set the wall |
| mapBarcode | Mark node as barcode node |
| cellVisted | Mark node as visited |
| setStart | Set the start of the maze and open the wall of the maze for entrance |
| setGoal | Set the end of the maze and open the wall of the maze for exit |
| printMap | Print the map, including the direction of the vehicle movement if any |
| printVistedMap | Print the map, where ‘X’ would mark that the node has been visited |
| initCar | Initialize the vehicle |
| moveCar | Move the current coordinate of the car |
| setCarStart | Set the starting coordinates of the car |
| explore\_map | Uses Depth First Search to explore the entire map, it will check for north, east, south and west for the wall respectively and moving to the node that it first scanned that isn’t visited and there is no wall blocking.  This will recursively occur until there is no more available node to visit. |
| assign\_cost\_cell | Uses Breath First Search to assign the cost of the map from start to goal. It will add the cost of the neighboring node that has not been visited and is not being blocked by a wall. |
| get\_shortest\_path | Starts from the goal and keeps taking the node that goes in lower value until it reaches the start. It will queue all the values in a doubly linked list so that the coordinates for traversing is bi directional. |

**Motor**

|  |  |
| --- | --- |
| **Function** | **Use** |
| init\_gpio\_motor | Initialize the GP pins required for motor |
| init\_motor\_a | Set the default pulse width modulation value for motor driver a |
| init\_motor\_b | Set the default pulse width modulation value for motor driver b |
| control\_wheels | Set the wheel speed and direction based on user selection of duty cycle and direction |
| user\_selection\_task | Ask the user to choose the duty cycle and direction of travel repeatedly |

**Ultrasonic**

|  |  |
| --- | --- |
| **Function** | **Use** |
| init\_gpio | Initialize the GP pins required for ultrasonic |
| calculate\_distance | Based on time provided, calculate the distance travelled |
| handle\_sound\_interrupt | Upon rising and falling edge, start or deduce time taken to send and receive ultrasonic sound and send it to calculate\_distance |
| generate\_sound\_task | Generate Ultrasonic sound every second for handle\_sound\_interrupt to trigger |
| print\_distance | Print the distance based on the calculated distance |

**Wifi**

|  |  |
| --- | --- |
| **Function** | **Use** |
| ssi\_init | Initialise SSI handler |
| cgi\_init | Initialise CGI handler |
| ssi\_handler | Displays car/pico data on web server and serial monitor |
| cgi\_handlers | Links html requests/routes to function triggers |