Algorithm Description Document (ADD) for Palomar CSCI 212 FINAL using osoyoo robot

Version 1.2

Adam Amer
Timothy Liu
Cody McKinney
Nathan White

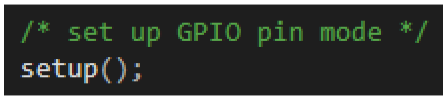
GROUP #3
05.29.2021

setup()

Description

setup() sets up the Raspberry Pi GPIO pins by configuring specified pins to behave either as an input or output. On method call, both the left and right motors are initialized as output devices. The sonic sensor trigger is initialized as an output device. The sonic sensor echo is initialized as an input device.

Syntax



```
/* set up GPIO pin mode */
setup();
```

Returns

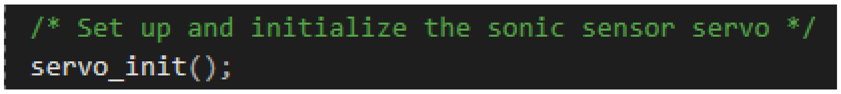
Nothing.

servo_init()

Description

servo_init() initializes the sonic sensor servo. On method call, the GPIO manually writes ticks to the pins representing the sonic sensor servo motor. The sensor will turn left, center, right, and end on center to show the servo has been initialized.

Syntax



```
/* Set up and initialize the sonic sensor servo */
servo_init();
```

Returns

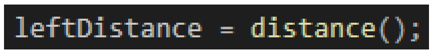
Nothing.

distance()

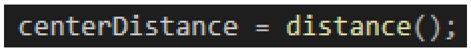
Description

distance() is used to measure the distance from the sonic sensor to an object. On method call, the sonic sensor trigger is briefly enabled. The time it takes to send and return the echo is calculated and converted centimeters.

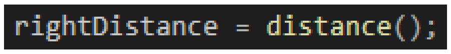
Syntax



```
leftDistance = distance();
```



```
centerDistance = distance();
```



```
rightDistance = distance();
```

Call distance() to use the sonic sensor to determine a distance between the robot and an object.

Returns

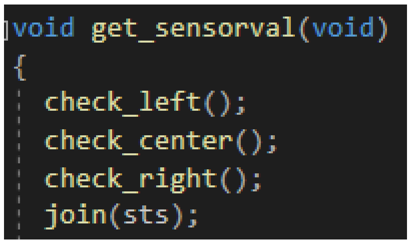
distance() returns an integer value associated with the distance calculated from the sonic sensor's echo.

check_left()

Description

check_left() is used to determine if an obstacle is within range on the left side of the robot. On method call, the sonic sensor servo motor moves to the left position, calls distance() function, and determines if the distance between the robot and an object is too close. A character array is then altered between 0 and 1.

Syntax



```
void get_sensorval(void)
{
    check_left();
    check_center();
    check_right();
    join(sts);
```

Call check_left() to determine the distance from the robot to an object on the left side.

Returns

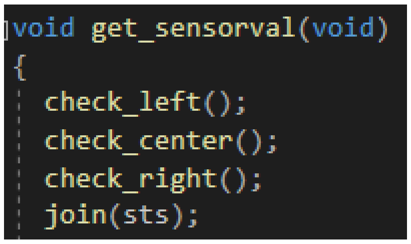
Nothing. But on method call, check_left() will check the left distance and change the sensor array value for the left direction to 0 or 1.

check_center()

Description

check_center() is used to determine if an obstacle is within range in front of the robot. On method call, the sonic sensor servo motor moves to the center position, calls distance() function, and determines if the distance between the robot and an object is too close. A character array is then altered between 0 and 1.

Syntax



```
void get_sensorval(void)
{
   check_left();
   check_center();
   check_right();
   join(sts);
```

Call check_center() to determine the distance from the robot to an object in front of the robot.

Returns

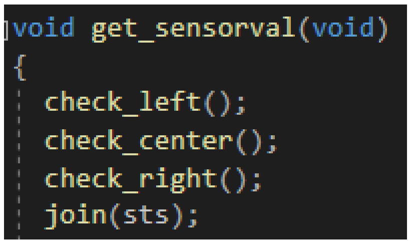
Nothing. But on method call, check_center() will check the forward distance and change the sensor array value for the center direction to 0 or 1.

check_right()

Description

check_right() is used to determine if an obstacle is within range on the right side of the robot. On method call, the sonic sensor servo motor moves to the right position, calls distance() function, and determines if the distance between the robot and an object is too close. A character array is then altered between 0 and 1.

Syntax



```
void get_sensorval(void)
{
  check_left();
  check_center();
  check_right();
  join(sts);
```

Call check_right() to determine the distance from the robot to an object on the right side.

Returns

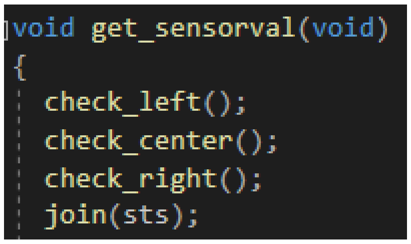
Nothing. But on method call, check_right() will check the right distance and change the sensor array value for the right direction to 0 or 1.

join()

Description

join() is used to create the sensorval string. On method call, the character ('0' or '1') for all 3 directions is concatenated to form a 3 character string to compare.

Syntax



```
void get_sensorval(void)
{
  check_left();
  check_center();
  check_right();
  join(sts);
```

join() is called after checking all 3 directions. The characters are joined into one string to compare.

Returns

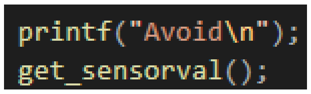
Nothing. Builds a string from 3 independent characters.

get_sensorval()

Description

get_sensorval() calls all 3 directional check_direction() functions and uses the join() function to create the sensorval string to compare. Sensorval is used to determine a direction to move the robot.

Syntax



```
printf("Avoid\n");
get_sensorval();
```

get_sensorval() is called to determine the robot's freedom to move by checking the three forward directions. The get_sensorval() function is used to either avoid an obstacle or follow an object.

Returns

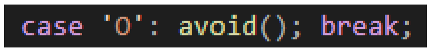
Nothing.

avoid()

Description

avoid() checks all 3 directions for an object and concatenates a string to determine objects detected within range. A series of 'if' statements handle the various directions and choose a movement (left, right, forward) for the robot to move in order to avoid detected objects in the vicinity.

Syntax



```
case 'O': avoid(); break;
```

avoid() is called when the UDP input is 'O', representing the "Obstacle Avoid" button on the osoyoo iPhone and Android app as referenced in the Interface Control Document (ICD).

Returns

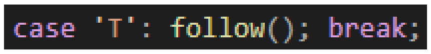
Nothing.

follow()

Description

follow() on method call checks all 3 directions for an object and concatenates a string to determine objects detected within range. A series of ‘if’ statements handle the various directions and choose a movement (left, right, forward) for the robot to move in order to follow another object. The robot will choose to head towards whatever object is closest in an attempt to follow a leader robot.

Syntax



```
case 'T': follow(); break;
```

follow() is called when the UDP input is ‘T’, representing the “Line Tracking” button on the osoyoo iPhone and Android app as referenced in the Interface Control Document (ICD).

Returns

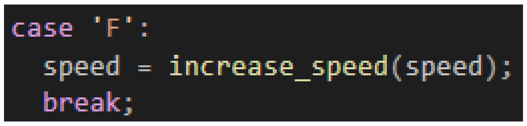
Nothing.

increase_speed()

Description

increase_speed() increases the movement speed of the robot. On method call, an integer speed is taken as an input. The current speed value is increased by 200 and the new speed value is returned to the calling function. The function will cap at a maximum speed pwm frequency.

Syntax



```
case 'F':
  speed = increase_speed(speed);
  break;
```

increase_speed() is called when the UDP input reads F, representing the ‘F1’ button on the osoyoo iPhone and Android app as referenced in the Interface Control Document(ICD).

Parameters

increase_speed() takes an integer speed as input to alter.

Returns

increase_speed() returns an integer of the new speed.

decrease_speed()

Description

decrease_speed() decreases the movement speed of the robot. On method call, an integer speed is taken as an input. The current speed value is decreased by 200 and the new speed value is returned to the calling function. The integer will cap at a minimum speed pwm frequency.

Syntax

```
case 'I': speed = decrease_speed(speed); cur_status = p; break;
```

decrease_speed() is called when the UDP input reads ‘I’, representing the ‘F4’ button on the osoyoo iPhone and Android app as referenced in the Interface Control Document(ICD).

Parameters

decrease_speed() takes an integer speed as input to alter.

Returns

decrease_speed() returns an integer of the new speed.

udp_receiver()

Description

udp_receiver() takes a buffered UDP input and copies it to current status and pre-status character arrays. The current status (or current button received) is printed to the screen.

Parameters

udp_receiver() takes a character p, which is a buffered UDP input from a cell phone.

Syntax

```
char udp_receiver(void);
```

```
pthread_create (&th1, NULL, (void*)udp_receiver, NULL);
```

Create a thread to run the udp_receiver and udp_handler at the same time.

Returns

udp_receiver() returns a pre-status, which is the most previously input character.

udp_handler()

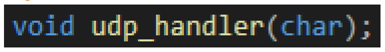
Description

udp_handler() takes a buffered UDP input and handles the character input as a command to determine which instruction to execute for the robot. The handler and the receiver run simultaneously and continuously until ctrl+c is entered from the terminal.

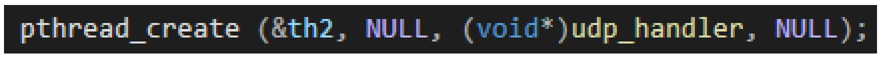
Parameters

udp_handler() takes a character p, which is the pre-status, or most recently pressed button. Current-status used to determine which key was last pressed.

Syntax



```
void udp_handler(char);
```



```
pthread_create (&th2, NULL, (void*)udp_handler, NULL);
```

Create a thread to run the udp_receiver and udp_handler at the same time.

Returns

Nothing.