Direction Finding for 2.4 GHz

Direction finding is a technique within RF-communication that uses different applications for reception of radio waves to determine the direction of the beacon. By measuring the signals in an antenna array in phase and anti-phase, the sum-difference between the respective signals can be used to detect beacons by rotating the antenna in the azimuth- and elevation plane. The goal for the project is to design and construct an antenna array with the applicable characteristics, design and dimension the robot arm holding and orienting the receiver antenna and develop a program code for the search algorithm and data processing. The application utilizes the Bluetooth Low Energy as the network protocol and operates on 2.4 GHz. By researching the existing literature within the field of science, a functioning direction finding prototype is going to be developed. Through empirical testing and observation, the limits and constraints are going to be derived from the given system.

Generated by Doxygen 1.9.3

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 data_a Struct Reference

Encoder values for zero-points This struct stores the encoder values for the zero-points.

`#include <data_processor.h>`

Data Fields

- int16_t **azimuth**
- int16_t **elevation**

3.1.1 Detailed Description

Encoder values for zero-points This struct stores the encoder values for the zero-points.

The documentation for this struct was generated from the following file:

- data_processor.h

3.2 data_s Struct Reference

Values measured at one point. This struct is used to store the measured values for encoder, RSSI at delta (phase-shifted) and RSSI for zigma(phase) at each measuring point.

`#include <data_processor.h>`

Data Fields

- int16_t **encoder**
- int16_t **delta**
- int16_t **zigma**

3.2.1 Detailed Description

Values measured at one point. This struct is used to store the measured values for encoder, RSSI at delta (phase-shifted) and RSSI for zigma(phase) at each measuring point.

The documentation for this struct was generated from the following file:

- data_processor.h

Chapter 4

File Documentation

4.1 buttons.h

```
1 #include <dk_buttons_and_leds.h>
2 #include <drivers/gpio.h>
3 #include "search.h"
4 #include "laser.h"
5
12 int configure_dk_buttons_leds();
```

4.2 data_processor.h

```
1 #include "encoder.h"
2 #include <zephyr/types.h>
3
9 #define MIN_VALID_RSSI -90
15 #define MAX_VALID_RSSI -25
22 typedef struct data_s
23 {
24     int16_t encoder;
25     int16_t delta;
26     int16_t zigma;
27 } matrix_x3;
28
34 typedef struct data_a
35 {
36     int16_t azimuth;
37     int16_t elevation;
38 } zeros;
39
50 void send_data(int16_t rssi, int index, int state);
51
58 void set_average_counter(int16_t value);
59
66 int16_t get_average(int16_t list[]);
67
75 void get_data(matrix_x3 *buffer_data, int N);
76
85 void value_validater(matrix_x3 *raw_data, int16_t *n);
86
95 void update_matrix(matrix_x3 *data, int16_t *n);
96
109 bool zero_point_validater(int16_t value_zigma, int16_t value_delta, int16_t ZIGMA_ZERO_VALUE);
110
119 int find_zero_point(matrix_x3 validated_values[], int n);
120
129 int16_t find_zigma_zero_value(matrix_x3 values[], int n);
130
```

4.3 encoder.h

```
1 #include <logging/log.h>
```

```
2 #include "servo.h"
3 #include <nrfx_qdec.h>
4
5
9 #define pin_a_azimuth 26
13 #define pin_b_azimuth 27
17 #define pin_a_elevation 30
21 #define pin_b_elevation 31
25 #define starting_angle_azimuth 0
29 #define starting_angle_elevation 0
30
39 int init_encoder_servos();
40
47 int init_encoder_azimuth();
48
55 int init_encoder_elevation();
56
63 void update_encoder(int N);
64
74 void angle_slow_move(int N, uint32_t angle);
75
82 int16_t get_encoder(int N);
83
```

4.4 initiater.h

```
1 #include "buttons.h"
11 int initiate_modules();
```

4.5 laser.h

```
1 #include <zephyr.h>
2 #include <sys/printk.h>
3 #include <device.h>
4 #include <devicetree.h>
5 #include <drivers/gpio.h>
6
12 int laser_init();
13
18 void laser_toggle(void);
19
25 void laser_set(int state);
```

4.6 observer.h

```
1 #include <bluetooth/bluetooth.h>
2 #include "data_processor.h"
3
4
5
17 int add_filter_accept_list_from_string(const char *addr_str,const char *type);
18
28 int init_bluethooth_scan();
29
44 static void device_found(const bt_addr_le_t *addr, int8_t rssi, uint8_t type, struct net_buf_simple *ad);
45
51 void set_observer(bool state);
52
53
54
55
56
57
58
59
60
61
```

4.7 search.h

```
1 #include "observer.h"
```

```
2
3 #define MOTOR_TEST 0
4
10 #define SEARCH_AZIMUTH 1
11 /*0
12  * @brief Activates serach in elevation
13  * 0 to deactivate search in elevation, 1 to activate search in elevation
14  */
15 #define SEARCH_ELEVATION 1
20 #define FINE_ACTIVATE 1
21
22
36 void sweep_search(int state, int16_t min_encoder_search, int16_t max_encoder_search, int increment);
37
46 int get_readings(matrix_x3 *main_readings, int16_t *n);
47
68 int16_t fine_sweeper(int state, int threshold_degrees, int threshold_search, int sweep_sector, int16_t
       zero_point);
69
75 void reset_readings();
76
77
90 zeros coarse_search();
91
102 zeros fine_search(zeros enc_values);
103
113 void validate_servo_zero_moved(int N, uint32_t zero_point_servo_angle);
114
121 void azimuth_servo_thread(uint32_t *azimuth_thread_servo_angle);
122
129 void elevation_servo_thread(uint32_t *elevation_thread_servo_angle);
```

4.8 servo.h

```
1 #include <zephyr.h>
2 #include <logging/log.h>
3 #include <drivers/gpio.h>
4
8 #define servo_azimuth_N 0
12 #define servo_azimuth_pin 2
16 #define servo_elevationl_N 1
20 #define servo_elevation_pin 3
24 #define servo_antenna_N 2
28 #define servo_antenna_pin 4
29
34 int timer_init();
35
40 int timer_start();
41
51 int servo_init(uint32_t N, int servo_pin);
52
59 void angle_move_servo(int N, uint32_t angle);
60
80 void raw_move_servo(int N, uint32_t position);
81
82
83 uint32_t sin_scaled(uint32_t input, uint32_t input_max, uint32_t output_min, uint32_t output_max);
84
91 void increment_servo(int N);
92
99 void decrement_servo(int N);
100
108 int16_t get_servo_angle(int N);
109
110
111
```

Index