Embedded System Lab

Final Acceptance Presentation

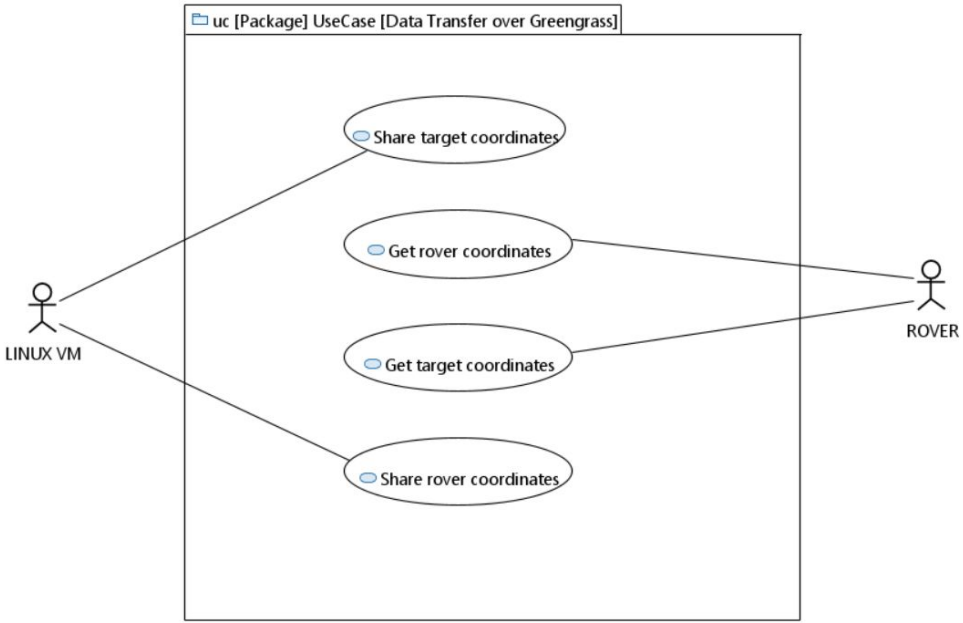
Group 12

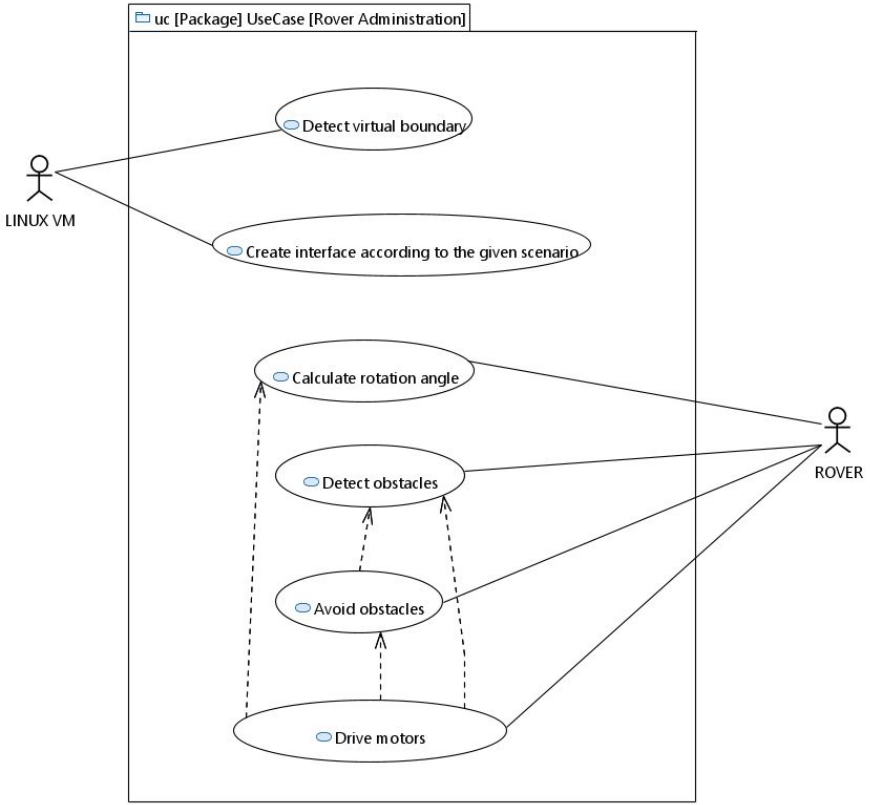
1. Namik Mert Tuncbilek - 7213712
2. Nijat Dashdamirov - 7213892
3. Vignesh Arumugam - 7213710

Intro to the Target finder

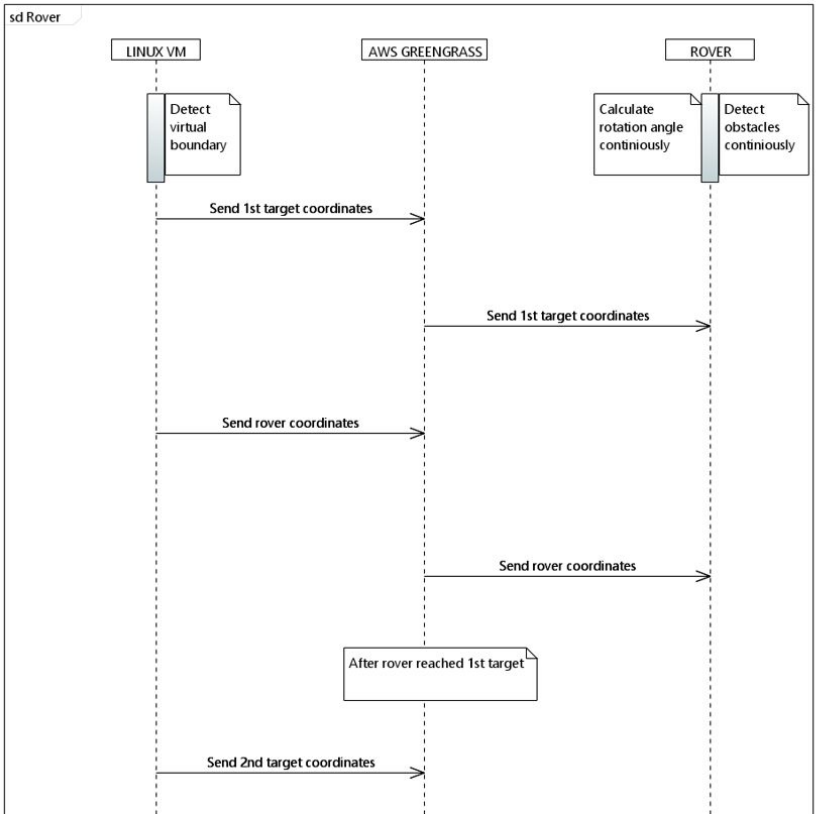
Language	:	CPP
Lines of Code	:	300
Board	:	ESP32
Algorithm/Formula used	:	Slope of the line; Graph coordinate plane; Radian to degree;
Cloud	:	AWS
Edge computing	:	AWS IoT Greengrass
Speciality	:	Only Internet is enough; Not required of AWS instance to run

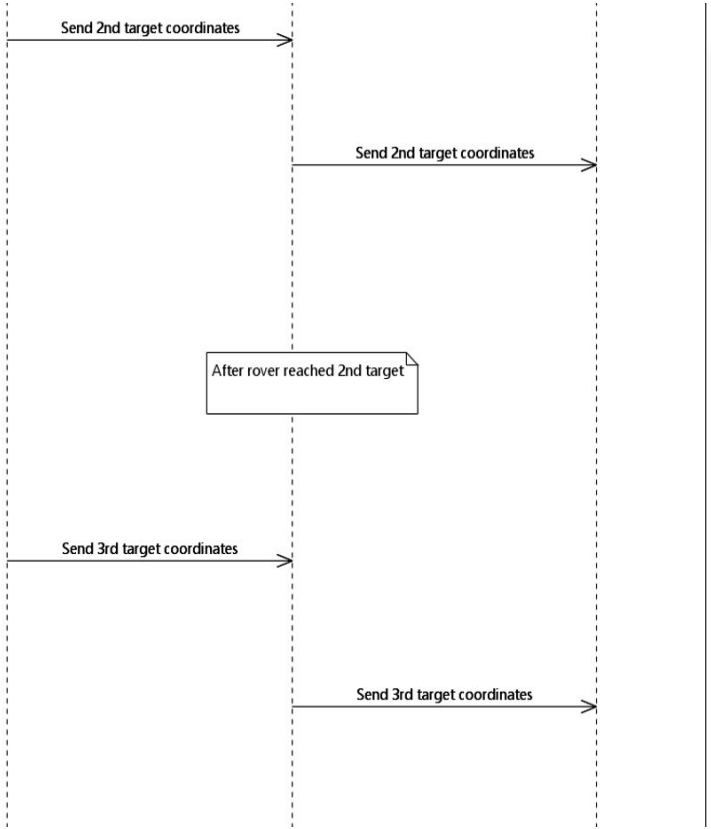
Use case Diagram



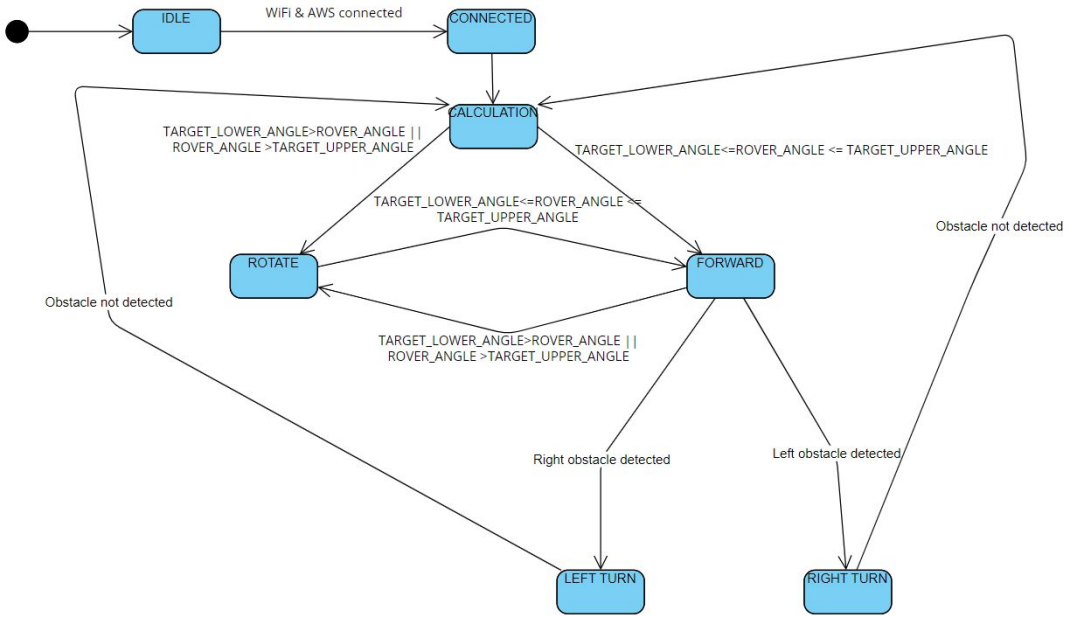


Sequence Diagram





State Diagram



Code Walk Through - Initial

```
#include <Arduino.h>
#include <cmath>
#include "motorDriver.h"
#include "sensorDriver.h"
#include "AWS.h"
#include "parsedData.h"

void taskOne( void * parameter);     //  LED
void taskTwo( void * parameter);     //  AWS connection
void taskThree( void * parameter);   //  Sensor
void taskFour( void * parameter);    //  Motor

enum Side
{
  Right = 0,   /* Rover need to turn RIGHT, because obstacle is at Left side */
  Left = 1   /* Rover need to turn LEFT, because obstacle is at Right side */
};

/*Angle of rover pointing towards Target with tolerance upper value*/
static int16_t angle_upper;
/*Angle of rover pointing towards Target with tolerance lower value*/
static int16_t angle_lower;
/*Degree between the current rover angle to the target point*/
static double degree;
/*Obstacle detected status variable*/
static boolean obstacle_detected =  false;
/*Variable having the current side where rover need to turn to avoid obstacle*/
static Side rover_side;


```

CWT - Continuation - Task 1

```
88  void taskOne( void * parameter )
89  {
90      //example of a task that executes for some time and then is deleted
91      for(;;)
92      {
93        // Serial.print("\nHello from task 1");
94
95        //Switch on the LED
96        digitalWrite(LED_BOARD, HIGH);
97        // Pause the task for 1000ms
98        delay(100); //This delay doesn't give a chance to the other tasks to execute
99        //vTaskDelay(100 / portTICK_PERIOD_MS); //this pauses the task, so others can execute
100       // Switch off the LED
101       digitalWrite(LED_BOARD, LOW);
102       // Pause the task again for 500ms
103       vTaskDelay(1000 / portTICK_PERIOD_MS);
104
105     }
106     Serial.println("Ending task: 1");
107     vTaskDelete( NULL );
108 }
109
110
```

CWT - Continuation - Task 2

```
111  void taskTwo( void * parameter )
112  {
113
114      for(;;)
115      {
116        Serial.print("\n");
117
118        awsobj.stayConnected();
119
120        vTaskDelay(30 / portTICK_PERIOD_MS);
121
122      }
123      Serial.println("Ending task: 4");
124      vTaskDelete( NULL );
125
126
127  }
128
```

CWT - Continuation - Task 3

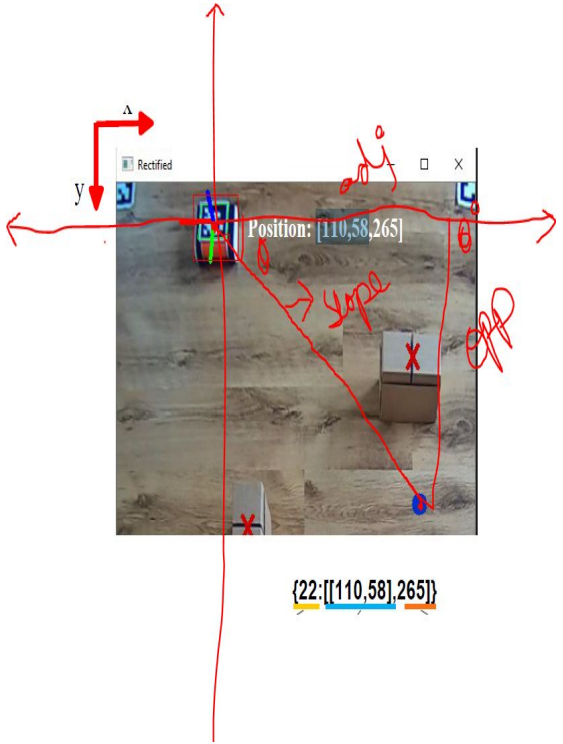
```
129  void taskThree( void * parameter )
130  {
131
132    int16_t* arr;
133
134      for(;;)
135      {
136        Serial.print("\n");
137
138        arr = sensorobj.reading();
139
140      Serial.print("X = ");
141      Serial.print(arr[0]);
142      Serial.print(" Y = ");
143      Serial.print(arr[1]);
144      Serial.print(" Z = ");
145      Serial.print(arr[2]);
146
```

CWT - Continuation - Task 3

```
148
149         if((arr[0]<150) && (arr[0]>0))
150         {
151           /*Turn right*/
152
153           rover_side = Right;
154           obstacle_detected = true;
155
156         }
157         else if((arr[2]<150) && (arr[2]>0))
158         {
159           /*Turn left*/
160           rover_side = Left;
161           obstacle_detected = true;
162         }
163         else
164         {
165           obstacle_detected = false;
166         }
167
168
169       vTaskDelay(30 / portTICK_PERIOD_MS);
170
171       }
172       Serial.println("Ending task: 3");
173       vTaskDelete( NULL );
174   }
```

CWT – Continuation – Task 4[Degree Calc]

```
176  void taskFour( void * parameter )
177  {
178
179    for(;;)
180    {
181      /*Implementing slope formula of two points[(y2-y1)/(x2-x1)][target point & rover point]*/
182      short num = target_y - rover_y;
183      short den = target_x - rover_x;
184      double slope = (double)num/(double)den;
185
186      /*Applying tan inverse formula to find the angle between target point and rover direction*/
187      double radian = atan(slope);
188      /*Converting radian to degree*/
189      degree = (radian*180)/3.1415;
190
191    /*If degree is in negative changing that into positive value*/
192    if(degree < 0)
193    {
194      degree = ((double)-1)* degree;
195    }
196    else
197    {
198      /*DO nothing*/
199    }
200
201
```



CWT – Continuation – Task 4[Quick Turn]

```
202     /*Divided the whole area into four coordinates w.r.t rover and target position*/
203     /*This will make the rover to change its direction towards target within minimum rotation*/
204     if((target_x > rover_x)&&(target_y>rover_y))
205     {
206       /*4th coordinate*/
207       degree = (double)360 - degree;
208     }
209     else if((target_x>rover_x)&&(rover_y>target_y))
210     {
211       /*Do nothing since target is in 1st coordinate*/
212     }
213     else if((rover_x>target_x)&&(rover_y>target_y))
214     {
215       /*2nd coordinate*/
216       degree = (double)180 - degree;
217     }
218     else if((rover_x>target_x)&&(target_y>rover_y))
219     {
220         /*3rd coordinate*/
221         degree = (double)180 + degree;
222     }
223     else
224     {
225
226     }
227
228     /*Applying 30 degree tolarence to the angle*/
229     angle_upper = (int16_t)degree + 30;
230     angle_lower = (int16_t)degree - 30;
231
```

CWT - Continuation - Task 4[NO Obs]

```
235         if(obstacle_detected != true)
236         {
237             if((rover_angle<=angle_upper)&&(rover_angle>=angle_lower))
238             {
239               /*Move forward*/
240               motorobject_motor.set_speed(MotorA, Backward, 250);
241               motorobject_motor.set_speed(MotorB, Forward, 250);
242             }
243             else
244             {
245                 if(degree<(rover_angle+180))
246                 { /*Spinning left*/
247                  motorobject_motor.set_speed(MotorA, Backward, 150);
248                  motorobject_motor.set_speed(MotorB, Backward, 150);
249                 }
250                 else
251                 { /*Spinning Right*/
252                  motorobject_motor.set_speed(MotorA, Forward, 150);
253                  motorobject_motor.set_speed(MotorB, Forward, 150);
254                 }
255             }
256         }
```

CWT - Continuation - Task 4[Obs Detected]

```
257         else
258         {
259           if(rover_side == Right)
260           {
261             /*Spinning Right*/
262             obstacle_detected = false;
263              motorobject_motor.set_speed(MotorA, Forward, 150);
264              motorobject_motor.set_speed(MotorB, Forward, 150);
265              delay(300);
266              /*Move forward*/
267              motorobject_motor.set_speed(MotorA, Backward, 250);
268              motorobject_motor.set_speed(MotorB, Forward, 250);
269              delay(200);
270
271           }
272           if(rover_side == Left)
273           {
274              /*Spinning Left*/
275              obstacle_detected = false;
276              motorobject_motor.set_speed(MotorA, Backward, 150);
277              motorobject_motor.set_speed(MotorB, Backward, 150);
278              delay(300);
279              /*Move forward*/
280              motorobject_motor.set_speed(MotorA, Backward, 250);
281              motorobject_motor.set_speed(MotorB, Forward, 250);
282              delay(200);
283           }
284
285         }
286
```

CWT - Continuation - AWS.cpp Initial

```
33  /* The MQTT topics that this device should publish/subscribe to */
34  #define AWS_IOT_SUBSCRIBE_TARGET_TOPIC "esp32/target"
35  #define AWS_IOT_SUBSCRIBE_ROVER_TOPIC "esp32/rover"
36  #define ROVER_VALUES_NUM 4
37  #define TARGET_VALUES_NUM 2
38
39  #define TARGET_X_COOR 0
40  #define TARGET_Y_COOR 1
41
42  #define ROVER_X_COOR 1
43  #define ROVER_Y_COOR 2
44  #define ROVER_ANGLE  3
45
46  #define MAX_NO_VALUES 3
47
48  int16_t target_x = 0;
49  int16_t target_y = 0;
50
63  void messageHandler(String &topic, String &payload)
64  {
65    StaticJsonDocument<200> doc;
66    boolean number_detected = false;
67    int16_t rover[ROVER_VALUES_NUM];
68    int16_t target[TARGET_VALUES_NUM];
69    String temp;
70    int16_t value = 0;
71    char store[MAX_NO_VALUES];
72    int16_t digit_num = 0;
73    deserializeJson(doc, payload);
```

CWT - Continuation - Message Handler Target

```
75  if(topic == "esp32/target")
76  {
77    temp = "target";
78
79    int num = 0;
80
81    const char* message = doc[temp];
82
83
84    for(;*message != '\0';*++message)
85    {
86      if((*message>=48)&&(*message<=57))
87      {
88        number_detected = true;
89        store[digit_num] = *message;
90        if(digit_num == 2)
91        {
92
93        }
94        else
95        {
96          store[digit_num + 1] = '\0';
97        }
98
99        digit_num += 1;
100     }
```

```
101       else
102       {
103         if(number_detected == true)
104         {
105           value = atoi(store);
106           target[num] = value;
107           num++;
108           digit_num = 0;
109           number_detected = false;
110         }
111         else
112         {
113           /*Do Nothing*/
114         }
115       }
116     }
117
118     target_x = target[TARGET_X_COOR];
119     target_y = target[TARGET_Y_COOR];
120
121   }
```

CWT - Continuation - Message Handler Rover

```
122     else if(topic == "esp32/rover")
123     {
124       temp = "rover";
125       int num = 0;
126       const char* message = doc[temp];
127
128
129       for(;*message != '\0';*++message)
130       {
131         if((*message>=48)&&(*message<=57))
132         {
133           number_detected = true;
134           store[digit_num] = *message;
135
136           if(digit_num == 2)
137           {
138
139           }
140           else
141           {
142             store[digit_num + 1] = '\0';
143           }
144
145           digit_num += 1;
146         }
```

```
147       else
148       {
149         if(number_detected == true)
150         {
151           value = atoi(store);
152           rover[num] = value;
153           num++;
154           digit_num = 0;
155           number_detected = false;
156         }
157         else
158         {
159           /*Do Nothing*/
160         }
161       }
162     }
163
164     rover_x = rover[ROVER_X_COOR];
165     rover_y = rover[ROVER_Y_COOR];
166     rover_angle = rover[ROVER_ANGLE];
167
168   }
```

Hurdles Went Through

- No topic received after few minutes [Task Delay of AWS should not be higher]
- Rover Stuck up suddenly, couldn’t know where to go [Network issue]
- Rover behaving improperly [Task Scheduling]
- Camera connectivity problem in VM [Configure USB of VM]
- Image Shape not captured & problem with Image recognition [Don’t run two Python commands at the same time, properly close the terminals, recheck the commands for typo error]

Thank You