Yuye Andreas Meier, 71610, 18INB-2

Dokumentation Aufgabenstellung Selbstortung

Versuche und Tests

IR-Empfänger

- Nutzung von LED7 and GPIO 12 als Infrarot Signal, das von Objekten reflektiert wird unter Tageslicht.
- ADC mit 12Bit Width (max. Wert 4095).

PT6 (Daylight Sensor) als Vorderer Sensor

	ADC Wert > 3000	ADC Wert = 4095
Plastikbehälter	Ab ca. 3cm	Ab ca. 1,5cm
Powerbank	Ab ca. 4cm	Ab ca. 2cm
Hand	Ab ca. 4cm	Ab ca. 2cm
Tshirt	Ab ca. 3cm	Ab ca. 1,5cm

PT 2 und 3 (seitliche Sensoren)

	ADC Wert > 3000	ADC Wert = 4095
Plastikbehälter	Ab ca. 2cm	Ab ca. 1cm
Powerbank	Ab ca. 3cm	Ab ca. 1.5cm
Hand	Ab ca. 3cm	Ab ca. 1.5cm
Tshirt	Ab ca. 2cm	Ab ca. 1cm

- PT6 über PT1 genutzt da unter LED7 PT1 permanent 4095 ausgibt.
- PT 4 und 5 nach hinten und unten in hier nicht nötig.

Yuye Andreas Meier, 71610, 18INB-2

Motoren PWM für Bewegung und Drehung

- PWM mit Auflösung von 10Bit und Frequenz von 50k Hz genutzt.
- Bei hohem duty cycle von 1023(max. Wert) sehr ungenaue Bewegung → niedrigere duty nötig, um Genauigkeit zu erhöhen.
- Ab ungefähr duty cycle von 300 bewegt sich der Roboter rückwärts bis gar nicht.
- Niedriger duty cycle von 300 ausgenutzt, um genaue Drehung des Roboters zu gewährleisten.
- Drehung und Bewegungsgeschwindigkeit des Roboters aufgrund von Vibrationsfortbewegung sehr stark von Bodengegebenheit und geeichten Bauteilen abhängig.
- 90° Drehung erfordert bei duty von 300 ca. 11Sekunden im Durchschnitt nach mehreren Tests.
- Relativ genaue Fortbewegung des Roboters bei duty von 400-600, da linke Seite des Roboters bei selber duty stärker die Bewegung beeinflusst hat wurde die duty des linken Motors schwächer und des rechten Motors stärker eingestellt.
- Fortbewegungsgeschwindigkeit von ungefähr 2cm pro Sekunde.

Yuye Andreas Meier, 71610, 18INB-2

Versuch von drahtloser Positionsausgabe über http Webserver

- Wifi connection erfolgreich.
- Webserver starten erfolgreich sowie Ausgabe von html code über GET request.
- Aktualisieren von Positionsdaten auf laufenden Server nicht erfolgreich hinbekommen ohne den Server zu schließen einen neuen GET handler an den Server zu übergeben und Daten zu ändern und danach wieder den Server zu starten → würde bei Echtzeit während Roboter sich orientieren muss den Algorithmus blockieren während auf Serverstart gewartet wird.

Versuche von Algorithmen

- Interpretation der Aufgabe:
  - Roboter bewegt sich auf gerader Linie, bis er auf Hindernis stößt
  - Er umgeht das Hindernis und geht dann wieder auf die gerade Ursprungsstrecke
- Umsetzung:
  - Roboter fährt gerade aus bis Hindernis im Vorderen Sensor erkennt
  - Je nachdem welcher seitliche Sensor einen höheren Wert anzeigt wird in die gegenüberliegende Seite das Hindernis umfahren
  - Seitlicher Sensor trackt das Hindernis so lange bis es nicht mehr erkannt also umfahren wurde
  - Roboter dreht sich in die Ursprungslage zurück, um den linearen Pfad zu verfolgen
- Probleme der Umsetzung:
  - Sensorenreichweite zu gering um Objekte zuverlässig zu tracken
  - Drehung und Fortbewegung je nach Umgebungsbedingungen zu ungenau um Roboter gezielt auf geraden Pfad zu halten (Fehlen von Winkelsensoren usw.)

Finale Realisierung des Projekts

PWM der Motoren

```
//MOTOR RIGHT
//1023 max, 512Half
void motor_set_speed(uint16_t speed) {
    // Set duty cycle of LEDC channel
    ledc_set_duty(LEDC_HIGH_SPEED_MODE, LEDC_CHANNEL_0, speed);
    ledc_update_duty(LEDC_HIGH_SPEED_MODE, LEDC_CHANNEL_0);
}

//MOTOR LEFT
void motor_set_speed2(uint16_t speed) {
    // Set duty cycle of LEDC channel
    ledc_set_duty(LEDC_HIGH_SPEED_MODE, LEDC_CHANNEL_1, speed);
    ledc_update_duty(LEDC_HIGH_SPEED_MODE, LEDC_CHANNEL_1);
}


//Robot turns right
void go_right(){
    motor_set_speed(300);
    motor_set_speed2(0);
}

//Robot turns left
void go_left(){
    motor_set_speed(0);
    motor_set_speed2(300);
}


//Robot stays still
void go_still(){
    motor_set_speed(0);
    motor_set_speed2(0);
}

//Robot goes straight
void go_straight(){
    motor_set_speed(500);
    motor_set_speed2(390);
}
```

Auslesen der PT Werte vom ADC

```
//GET PT Values
gpio_set_level(GPIO_NUM_16, 1);
phototransistor_value_PT6 = adc1_get_raw(ADC1_CHANNEL_7);
reset_PTvalues();

gpio_set_level(GPIO_NUM_25, 1);
phototransistor_value_left = adc1_get_raw(ADC1_CHANNEL_7);
reset_PTvalues();

gpio_set_level(GPIO_NUM_26, 1);
phototransistor_value_right = adc1_get_raw(ADC1_CHANNEL_7);
reset_PTvalues();
```

Einfacher Algorithmus zur Objektvermeidung

```
if(phototransistor_value_PT6 < 3500){
    if(tracking){
        gettimeofday(&start_time,NULL);
        tracking = false;
    }
    //go_straight();
}
else if(phototransistor_value_right > phototransistor_value_left){
    ESP_LOGI("Going","left");

    //go_left();
    calc_distance();
    change_direction('l');
    vTaskDelay(11000 / portTICK_PERIOD_MS);

}
else{
    ESP_LOGI("Going","right");

    //go_right();
    calc_distance();
    change_direction('r');
    vTaskDelay(11000 / portTICK_PERIOD_MS);
}
```

Positions- und Distanzberechnung

```
//sets the direction the robot is facing in relation to the starting point
void change_direction(char dir){
    if (direction == 0){
        if(dir == 'l'){direction++;}
        if(dir == 'r'){direction = 3;}
    }
    else if(direction == 4){
        if(dir == 'l'){direction = 1;}
        if(dir == 'r'){direction--;}
    }
    else{
        if(dir == 'l'){direction++;}
        if(dir == 'r'){direction--;}
    }

}
```

```
//calculates and logs the distance to the starting point
void calc_distance(){
    if(tracking == false){
    gettimeofday(&end_time, NULL);
    timer = (float)(end_time.tv_sec - start_time.tv_sec) + ((float)(end_time.tv_usec - start_time.tv_usec) / 1000000);
    }
    switch(direction){
        case 2:
        Movevector[1] = Movevector[1] + timer * 2;
        break;
        case 1:
        Movevector[0] = Movevector[0] + timer * 2;
        break;
        case 3:
        Movevector[0] = Movevector[0] - timer * 2;
        break;
        default:
        Movevector[1] = Movevector[1] - timer * 2;

    }
    distance = sqrt(Movevector[0]*Movevector[0]+Movevector[1]*Movevector[1]);

    timer=0;
    tracking = true;

    ESP_LOGI("Position     ", "Distanz: %.2f cm    Koordinate x: %.2f    Koordinate y: %.2f    Alte Richtung: %d"
    , distance, Movevector[0], Movevector[1], direction);
}
```

- Direction values in Relation zur Startposition:
  - 2 → Geradeaus
  - 3 → Links
  - 1 → Rechts
  - 0 und 4 → Hinten
- Wenn Roboter sich nach links dreht erhöht sich direction um +1 und wenn er sich nach rechts dreht -1
- Geht der Wert unter 0 so setzt sich der Wert auf 3, da 0 und 4 beide für „Hinten“ definiert sind, gleiches gilt falls 4 sich erhöht dann geht der Wert auf 1
- Distanzberechnung über Koordinaten [x,y] und dem Satz des Pythagoras
- Je nachdem in welche „direction“ der Roboter sich fortbewegt verändern sich die Werte auf dem xy-Koordinatensystem, wobei der Koordinatenursprung die Startposition des Roboters ist