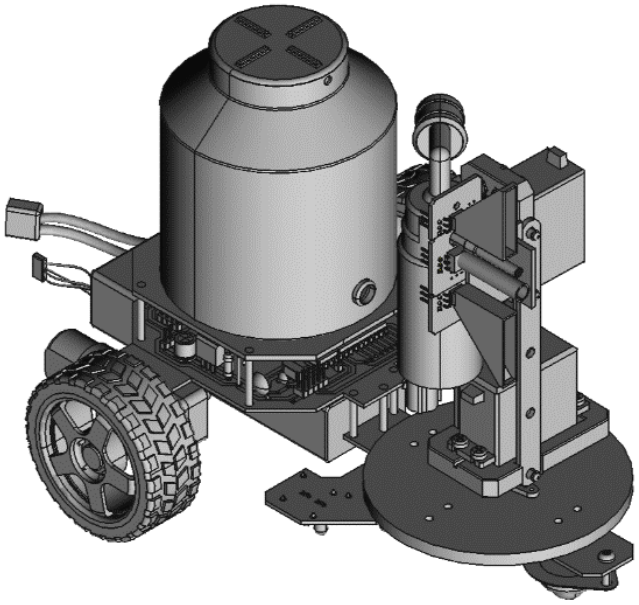
Nama : Rafli Zaki Rabbani
NIM : 2100415
Prodi : PTOIR

Robot Line Follower Pemadam Api

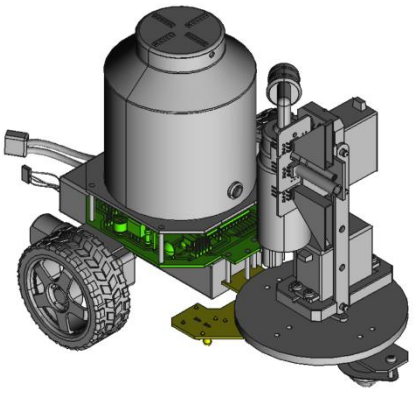


Gambar design robot

Penambahan subsistem pemadam api/kebakaran, pengembangan dari robot line follower yang pernah dibuat sebelumnya. Software yang digunakan:

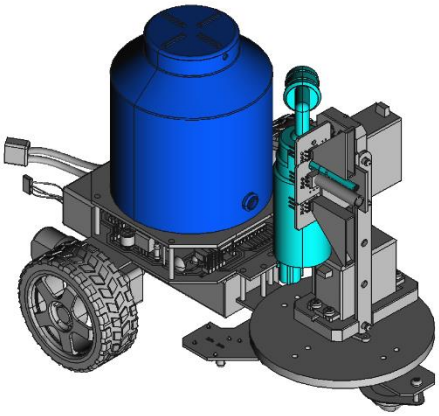
- FreeCAD : membuat design dan rangka robot
- KiCAD : membuat skematik rangkaian dan design PCB
- WinAVR, Programmer’s Notepad : coding/programming
- SimulIDE : simulasi

Design



Papan sirkuit utama (hijau), papan sirkuit
sensor line follower (kuning)

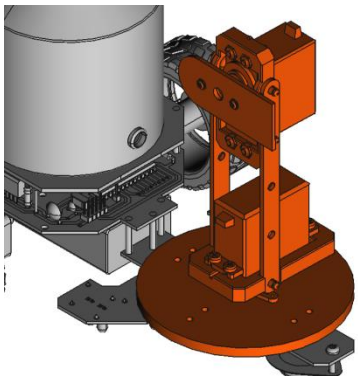
Bagian papan sirkuit utama adalah otak dari keseuruhan robot, dimana berisikan perangkat keras utama pemrosesan dan komponen lainnya tang diperlukan. Sistem robot berjalan menggunakan mikrokontroller Atmega328. Baterai Lipo yang digunakan sebagai sumber daya utama diletakkan dibawah papan sirkuit utama. Dua buah motor DC yang terletak disamping digunakan untuk memutar roda line follower. Dibagian depan-bawah dari papan sirkuit utama, terdapat papan sirkuit yang berisikan beberapa pasangan transmitter dan receiver infrared LED yang berfungsi untuk mendeteksi jalur line follower.



Water tank (biru), windshield water
pump (biru muda)

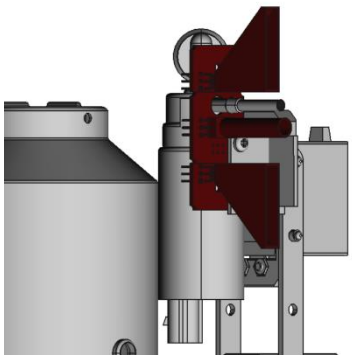
Pada sistem pemadam kebakaran digunakan tiga aktuator utama: pompa air menggunakan windshield water pump kecil yang bisa mengeluarkan semburan air hingga 3 meter, dua buah motor servo yang masing-masing mengendalikan posisi vertikal dan horizontal

dari sensor dan water pump. Mikrokontroler Atmega328 akan digunakan untuk mengelola sinyal "on" yang dikirim ke pompa air dan sinyal posisi/sudut yang dikirim ke servo.



Gambar penggerak utama pemadam
api dengan 2 buah servo motor

Untuk mendeteksi kebakaran, sistem menggunakan tiga sensor utama menggunakan sensor inframerah (infrared sensor): satu sensor tengah yang melihat langsung ke depan di mana water pump diarahkan, satu sensor yang melihat diagonal ke atas dari lokasi perkiraan water pump, dan satu sensor yang melihat diagonal ke bawah dari lokasi perkiraan water pump. Sensor tengah memiliki keputusan akhir untuk memadamkan api yang terdeteksi oleh salah satu sensor. Semua sensor memiliki bidang pandang (FOV) horizontal yang dibatasi menggunakan penghalang yang dibuat dengan 3D print, sehingga sensor dapat berfokus menganalisis bagian ruang vertikal. Sensor tengah memiliki bidang pandang yang dibatasi untuk berfokus ke satu titik didepan searah dengan arah semburan water pump.

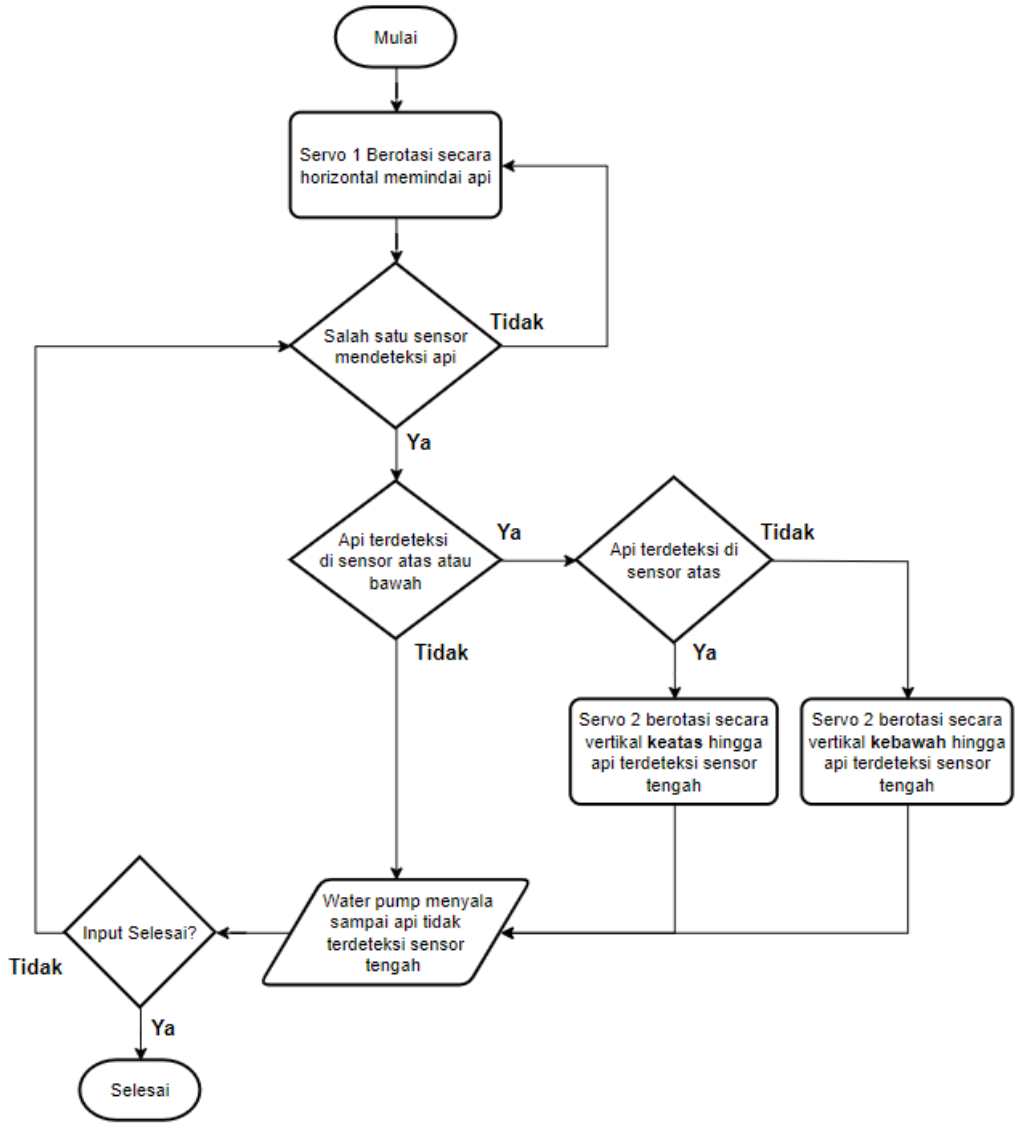


Gambar bagian sensor pemadam api

Gambar bidang pandang
ketiga sensor

Tujuannya adalah dengan menyapu sensor di sepanjang sumbu horizontal, sensor tengah hanya akan melihat titik di mana water pump diarahkan, sedangkan sensor atas dan bawah dapat memberitahu sensor tengah mengenai titik yang perlu diperiksa dengan sensor sentral. Saat menyapu, jika sistem mendeteksi sesuatu melalui sensor tengah, ia langsung

menganggapnya sebagai kebakaran dan mengaktifkan penyemprotan untuk memadamkannya. Jika mendeteksi sesuatu melalui sensor atas atau bawah, sistem menghentikan pemindaian horizontalnya dan mulai memindai secara vertikal ke arah sensor yang mendeteksi (atas atau bawah), memberi prioritas pada sensor bawah. Ketika rutinitas pemindaian vertikal selesai, sistem melanjutkan rutinitas pemindaian horizontalnya dan proses berulang. Gambar di bawah menggambarkan flowchart operasi dasar sistem pemadam api:

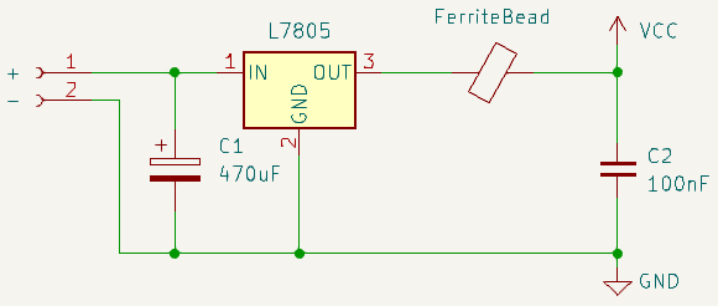


Flowchart sistem pemadam api

Hardware

1. Daya

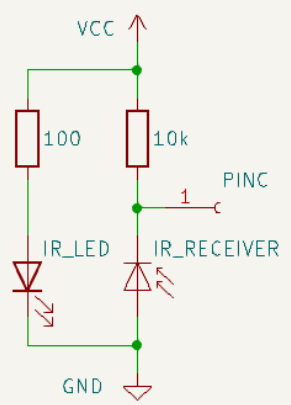
Sistem didukung oleh baterai Lipo 12 V yang digunakan untuk memberi daya pada papan sirkuit utama berisi Atmega328, motor DC servo, sensor, dan water pump. Untuk menurunkan tegangan menjadi 5V, digunakan rangkaian voltage regulator yang berisi LM7805 untuk menurunkan tegangan ke 5V, beserta induktor ferrite bead untuk memfilter noise frekuensi tinggi dan kapasitor 470uF dan 100nF untuk memfilter ripple. Gambar di bawah menunjukkan skema rangkaian regulator tegangan yang mencakup kapasitor input, output dan ferrite bead yang direkomendasikan dalam datasheet LM7805.



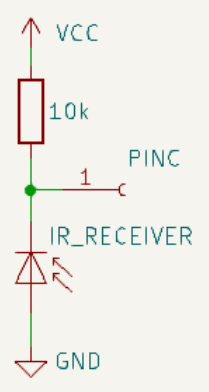
Regulator Tegangan

2. Sensor

Subsistem sensor menggunakan enam buah sensor utama: 3 sensor untuk subsistem line follower, 3 sensor untuk subsistem pemadam api. Sensor yang digunakan adalah fototransistor inframerah yang sensitif terhadap cahaya dalam rentang 940 nm. Pertimbangan desain yang penting untuk subsistem pemadam api adalah bahwa sensor yang digunakan harus cukup untuk membedakan api dari sumber cahaya lain yang bisa menjadi alarm palsu, dan pertimbangan desain terpenting untuk sensor adalah bahwa mereka memiliki bidang pandang yang luas terutama sensor atas dan bawah. Pertimbangan desain penting lainnya adalah mereka harus mampu mendeteksi target pada jarak tertentu agar sistem prototipe efektif.



IR LED transmitter dan receiver sebagai sensor line follower



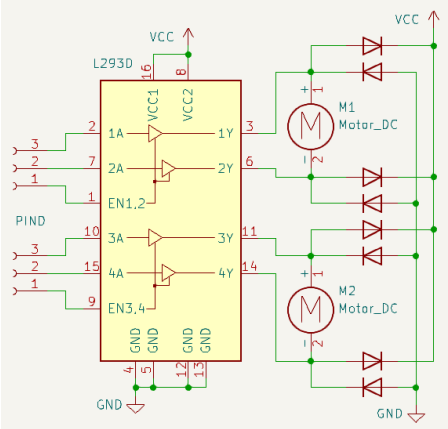
Hanya receiver sebagai pendeteksi api.

Rangkaian sensor dirangkai berdasarkan prinsip pembagi tegangan dengan resistor 10k ohm. Pada sensor subsistem line follower, dibuat pasangan-pasangan LED transmitter dan receiver yang dipasangkan berdekatan agar pemindaian pantulan cahaya dari jalur/line jelas. Hasil pembacaan nilai analog dari sensor bisa dihitung dengan:

$$V_{analog} = V_{in}\,\frac{R2}{(R1+R2)}, \qquad R1 = 10k, \qquad R2 = R\ \ sensor$$

3. Aktuator

Pada bagian subsistem line follower, dua buah roda pada bagian kanan dan kiri digerakkan menggunakan dua buah motor dc 3-6V. Motor dikontrol oleh motor driver L293D yang berfungsi mengisolasi daya motor dengan sinyal kontrol dari mikrokontroller. Arah putar motor ditentukan dengan menentukan pin 1A, 2A, 3A, dan 4A, sedangkan pengaktifan/on-off nya motor dikontrol oleh pin EN. Output dari motor driver yang tersambung ke motor diberi flyback diode yang berfungsi untuk melindungi motor driver dari lonjakan tegangan EMF balik.

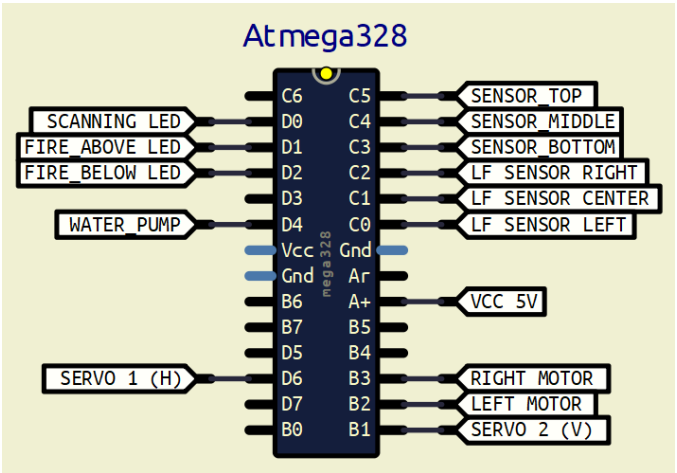


Rangkaian pada motor line follower

Selain itu, terdapat sebuah winshield water pump yang berisikan motor dc untuk menyemprotkan air, dan dua buah servo motor yang digunakan untuk menggerakkan dan mengarahkan sensor dan water pump. Servo motor yang digunakan menggunakan servo 5V yang memiliki sudut rotasi 180° untuk satu servo vertikal, dan 180° (atau 360°, opsional) untuk servo horizontal. Servo dikontrol dengan sinyal PWM (Pulse Width Modulation), dimana lebar sinyal tinggi dari PWM yang menentukan sudut servo, rentang lebar sinyal kontrol servo yang digunakan adalah 1ms sampai 2ms.

4. Mikrokontroller

Mikrokontroller yang digunakan untuk sistem ini adalah Atmega328, sebuah mikrokontroller CPU AVR 8-bit dengan clock speed 16 MHz. Berikut adalah gambar sambungan pin yang digunakan pada sistem yang dibuat:



Gambar sambungan pin yang digunakan serta labelnya

Atmega328 yang digunakan menggunkan 5V sebagai sumber daya dan tegangan referensinya. Untuk pembacaan sensor, Atmega328 memiliki ADC 10-bit, artinya ia dapat mengubah input tegangan analog menjadi salah satu dari 2^10 atau 1024 (0-1023) nilai pembacaan digital. ADC bisa dilkukan oleh pin C0 – C5 (total 6 ADC) jumlah yang pas untuk sistem yang dibuat. Nilai ADC bisa didapat dengan rumus:

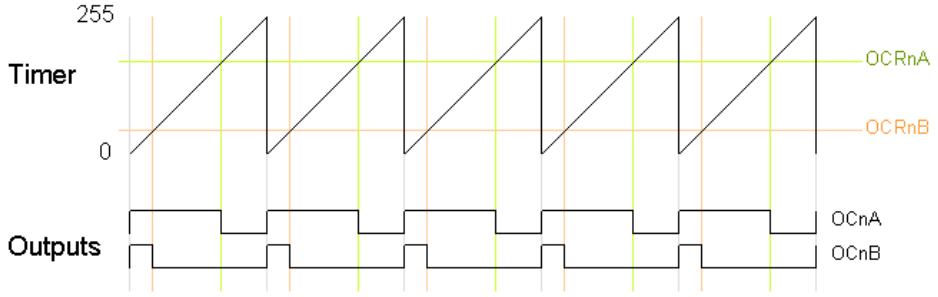
$$ADC = \left(\frac{V_{analog}}{A_{ref}}\right) 1023, \qquad \text{pada sistem ini } A_{ref} = VCC$$

Untuk mengontrol aktuator, pada water pump, motor untuk line follower dan beberapa LED indikator diatur dengan sinyal boolean (high dan low). Sedangkan servo diatur dengan lebar sinyal PWM (Pulse Width Modulation). Rentang lebar pulsa yang diterima servo untuk mengontrol dari sudut terendah hingga maksimum adalah 1ms sampai 2ms. Diketahui bahwa Atmega yang digunakan mempunyai frekuensi CPU 16 MHz, yang menghasilkan lebar PWM kecil. Untuk bisa mengontrol sudut servo, perioda yang dihasilkan setidaknya harus lebih dari 2ms, maka frekuensi maksimal PWM adalah:

$$f = \frac{1}{T} = \frac{1}{0.002} = 500\ Hz$$

Untuk memenuhi kebutuhan tersebut, kita bisa menggunakan konfigurasi berikut pada Timer/Counter yang akan digunkan:

- Fast PWM Mode
- Non-inverting
- Timer TOP value = 255 (default 8-bit)
- Timer prescaler = 256



Gambar PWM generation dengan Fast PWM Mode

Maka didapatkan:

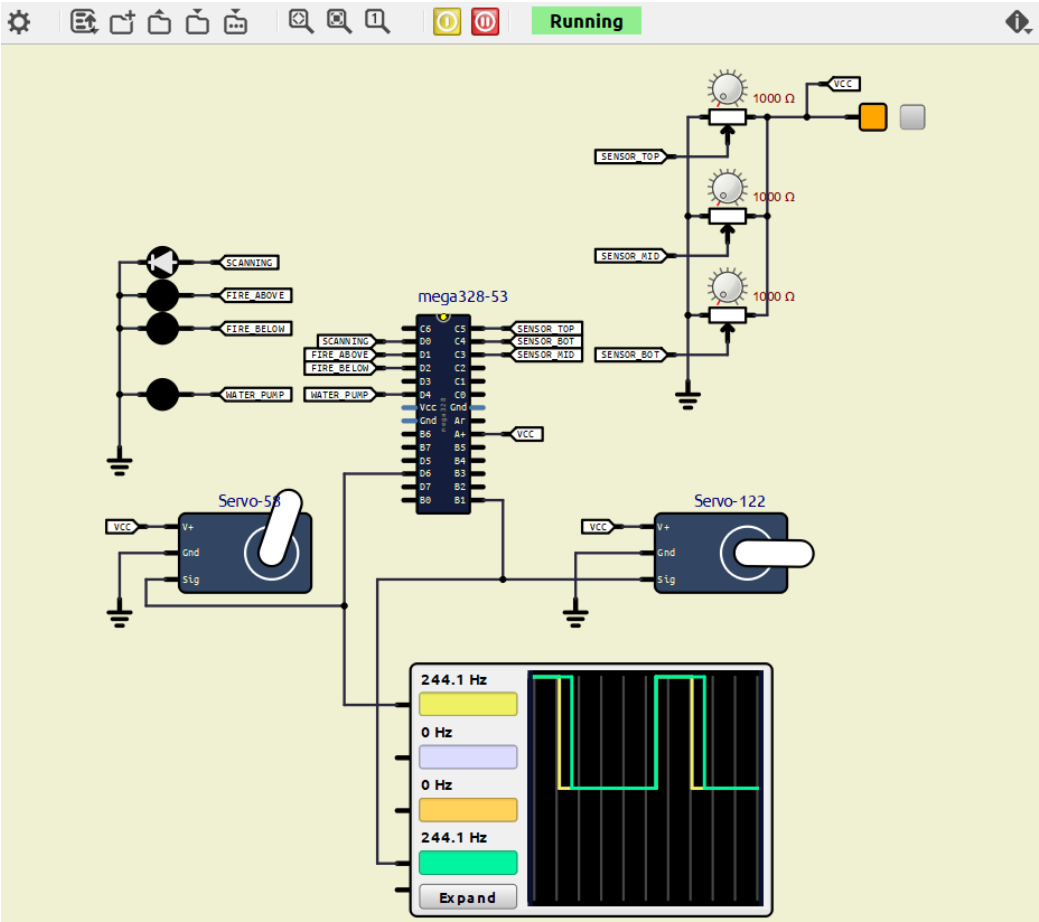
$$f_{timer} = \frac{F_{cpu}}{Prescaler} = \frac{16\ MHz}{256} = 62.5\ kHz$$

$$f_{pwm} = \frac{f_{timer}}{TOP + 1} = \frac{62.5\ kHz}{256} = 244.14\ Hz$$

Maka frekuensi PWM memenuhi frekuensi maksimal (500 Hz) untuk mengontrol servo. Perioda yang dihasilkan adalah:

$$T = \frac{1}{f} = \frac{1}{244.14} = 0.0041\ s = 4.1\ ms$$

Sehingga untuk mengontrol servo dengan rentang lebar pulsa 1 ms sampai 2 ms, kita hanya perlu mengontrol duty cycle sekitar 25% sampai 50% (dari 4.1 ms) dengan memotong sinyal timer dengan nilai OCRnA sepert yang terlihat pada gambar grafik Fast PWM Mode. Berdasarkan nilai top 255 dan pengurangan sedikit nilai karena perioda tidak bulat 4 ms dan disesuaikan dengan sudut servo yang digunakan, maka nilai OCRnA yang diperlukan terletak pada baris ke 13-15 pada code, dan konfigurasi PWM terletak pada baris 51-61.



Gambar simulasi subsistem pemadam api dengan SimulIDE

Dari osiloskop pada gambar bisa dilihat bahwa frekuensi PWM = 244.1 Hz dan lebar pulsa servo 1 (kuning) lebih kecil dari lebar pulsa servo 2 (hijau) membuat sudut putar servo 1 (kiri) lebih kecil dari servo 2 (kanan) yang berada pada 90 derajat.

Code :

```c
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdbool.h>

#define F_CPU 16000000UL

// batas deteksi sensor (sesuaikan dengan kalibrasi sensor)
#define FIRE_THRESHOLD 500
#define LINE_THRESHOLD 500

// sudut servo
#define DEGREE_180 125  //125 = 2 ms = 180 derajat
#define DEGREE_90 94    //93.8 = 1.5 ms = 90 derajat
#define DEGREE_0 62 //62.5 = 1 ms = 0 derajat

//deklarasi variabel
int PULSE_WIDTH = DEGREE_90;
bool line_following = true;
bool fire_searching = true;
bool fire_detected = false;
bool stop_aim = false;
bool fire_on_aim = false;
uint16_t adc_value[6];

// data enumerasi untuk lokasi api pada sensor
typedef enum {
    SENSOR_MID = 0,
    SENSOR_BOT,
    SENSOR_TOP
} Sensor;

Sensor fire_at = SENSOR_MID; // deklarasi, nilai awal

// inisiasi pin yang digunakan, input, output
void init_pin(void) {
    DDRD |= (1 << PD0) | (1 << PD1) | (1 << PD2) | (1 << PD3) | (1 << PD4) | (1 << PD5) | (1 << PD6);
    DDRC |= (1 << PC3) | (1 << PC4) | (1 << PC5);
    DDRC &= ~((1 << PC0) | (1 << PC1) | (1 << PC2));
    DDRB |= (1 << PB1) | (1 << PB2) | (1 << PB3);
}

// inisiasi pin c sebagai adc
void init_adc(void) {
    ADMUX = (1 << REFS0);
    ADCSRA = (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
    ADCSRA |= (1 << ADEN);
}

// inisiasi mode pwm
void init_pwm(void) {
    TCCR0A |= (1 << COM0A1) | (1 << WGM01) | (1 << WGM00); // Fast pwm mode, non-inverting pin OCR0A (D6)
    TCCR0B |= (1 << CS02); // prescaler frekuensi clock x256 pin OCR0A (D6)

    TCCR1A |= (1 << COM1A1) | (1 << WGM10); // Fast pwm mode, non-inverting pin OCR1A (B1)
    TCCR1B |= (1 << CS12) | (1 << WGM12); // prescaler frekuensi clock x256 pin OCR1A (B1)

    OCR0A = PULSE_WIDTH;

    OCR1A = PULSE_WIDTH;
}

// mengonversi input adc (pin c)
uint16_t adc_read(uint8_t adc_channel) {
    ADMUX = (ADMUX & 0xF8) | (adc_channel & 0x07);
    ADCSRA |= (1 <<ADSC);

    while (ADCSRA & (1 << ADSC));

    return ADC;
}

// membaca pin-pin adc
void adc_check(void) {
    for (int channel = 0; channel < 7; channel++) {
        adc_value[channel] = adc_read(channel);
    }
}

// Kontrol motor line follower
void motor_forward(void) {
    PORTB |= (1 << PB2);
    PORTB |= (1 << PB3);
}

void motor_left(void) {
    PORTB &= ~(1 << PB2);
    PORTB |= (1 << PB3);
}

void motor_right(void) {
    PORTB |= (1 << PB2);
    PORTB &= ~(1 << PB3);
}
```

```
95
96  void motor_stop(void) {
97      PORTB &= ~(1 << PB2);
98      PORTB &= ~(1 << PB3);
99  }
100
101 // cek keberadaan api pada sensor
102 void fire_check(void) {
103     if (adc_value[3] > FIRE_THRESHOLD) { // sensor tengah
104         fire_detected = true;
105         fire_at = SENSOR_MID;
106     } else if (adc_value[4] > FIRE_THRESHOLD) { // sensor bawah
107         fire_detected = true;
108         fire_at = SENSOR_BOT;
109     } else if (adc_value[5] > FIRE_THRESHOLD) { // sensor atas
110         fire_detected = true;
111         fire_at = SENSOR_TOP;
112     }
113 }
114
115 // merotasikan servo 1 (horizontal) dari sudut rendah ke tinggi (kiri)
116 void aim_left(int PULSE_WIDTH_NOW, int PULSE_WIDTH_RESULT) {
117     // loop sampai servo berada pada sudut maksimum atau api terdeteksi sensor
118     for (int i = PULSE_WIDTH_NOW; i < PULSE_WIDTH_RESULT && !fire_detected; i += 1) {
119
120         OCR0A = i;
121         adc_check();
122         _delay_ms(30);
123         PULSE_WIDTH = i;
124         fire_check();
125     }
126 }
127
128 // merotasikan servo 1 (horizontal) dari sudut tinggi ke rendah (kanan)
129 void aim_right(int PULSE_WIDTH_NOW, int PULSE_WIDTH_RESULT) {
130     // loop sampai servo berada pada sudut minimum atau api terdeteksi sensor
131     for (int i = PULSE_WIDTH_NOW; i > PULSE_WIDTH_RESULT && !fire_detected; i -= 1) {
132
133         OCR0A = i;
134         adc_check();
135         _delay_ms(30);
136         PULSE_WIDTH = i;
137         fire_check();
138     }
139 }
140
141 //
```

```
140
141 //
142 int main(void) {
143     init_pin();
144     init_pwm();
145     init_adc();
146     sei();
147
148     while (1) {
149
150         // reset variabel
151         OCR1A = DEGREE_90;
152         line_following = true;
153         fire_searching = true;
154         fire_on_aim = false;
155         stop_aim = false;
156
157         // reset indikator
158         PORTD &= ~(1 << PD0);
159         PORTD &= ~(1 << PD1);
160         PORTD &= ~(1 << PD2);
161         PORTD &= ~(1 << PD4);
162
163         // loop line follower
164         while (line_following) {
165
166             adc_check();
167
168             if (adc_value[1] > LINE_THRESHOLD) {
169                 motor_forward();
170             } else if (adc_value[0] > LINE_THRESHOLD) {
171                 motor_left();
172             } else if (adc_value[2] > LINE_THRESHOLD) {
173                 motor_right();
174             } else {
175                 motor_stop();
176                 line_following = false;
177             }
178
179             _delay_ms(50);
180         }
181
```

```
181         }
182         // loop servo 1 (horizontal), memindai api
183         while (fire_searching) {
184
185             PORTD |= (1 << PD0);
186             adc_check();
187
188             aim_left(PULSE_WIDTH, DEGREE_180);
189
190             if (fire_detected) {
191                 stop_aim = true; // agar posisi servo horizontal tetap, tidak rotasi ke kanan
192                 fire_searching = false;
193             }
194
195             if (!stop_aim) { // jika tidak terdeteksi api saat rotasi ke kiri, lanjut rotasi
196                 aim_right(PULSE_WIDTH, DEGREE_0);
197                 if (fire_detected) {
198                     fire_searching = false;
199                 }
200             }
201         }
202
203         PORTD &= ~(1 << PD0);
204
205         // loop servo 2 (vertikal), membidik dan memadamkan api
206         while (fire_detected) {
207
208             // jika api terdeteksi di sensor bawah
209             if (fire_at == SENSOR_BOT) {
210                 PORTD |= (1 << PD2);
211
212                 // servo 2 rotasi kebawah sampai api terdeteksi di sensor tengah
213                 for (int i = DEGREE_90; i > DEGREE_0 && !fire_on_aim; i -= 1) {
214
215                     OCR1A = i;
216                     adc_check();
217                     _delay_ms(30);
218
219                     if (adc_value[3] > FIRE_THRESHOLD) {
220                         fire_on_aim = true;
221                         fire_at = SENSOR_MID;
222                     }
223                 }
224
```

```
224
225                 // jika api terdeteksi di sensor bawah
226             } else if (fire_at == SENSOR_TOP) {
227                 PORTD |= (1 << PD1);
228
229                 // servo 2 rotasi keatas sampai api terdeteksi di sensor tengah
230                 for (int i = DEGREE_90; i < DEGREE_180 && !fire_on_aim; i += 1) {
231
232                     OCR1A = i;
233                     adc_check();
234                     _delay_ms(30);
235
236                     if (adc_value[3] > FIRE_THRESHOLD) {
237                         fire_on_aim = true;
238                         fire_at = SENSOR_MID;
239                     }
240                 }
241             }
242
243             // api terdeteksi di sensor tengah
244             if (fire_at == SENSOR_MID) {
245
246                 PORTD |= (1 << PD4); // water pump on
247
248                 // loop, selesai sat api tidak terdeteksi lagi di sensor tengah
249                 while (fire_on_aim) {
250
251                     _delay_ms(100);
252                     adc_check();
253                     if (adc_value[3] < FIRE_THRESHOLD) {
254                         fire_on_aim = false;
255                     }
256
257                 }
258
259                 PORTD &= ~(1 << PD4);  // water pump off
260                 fire_detected = false;
261             }
262         }
263     }
264 }
265
```