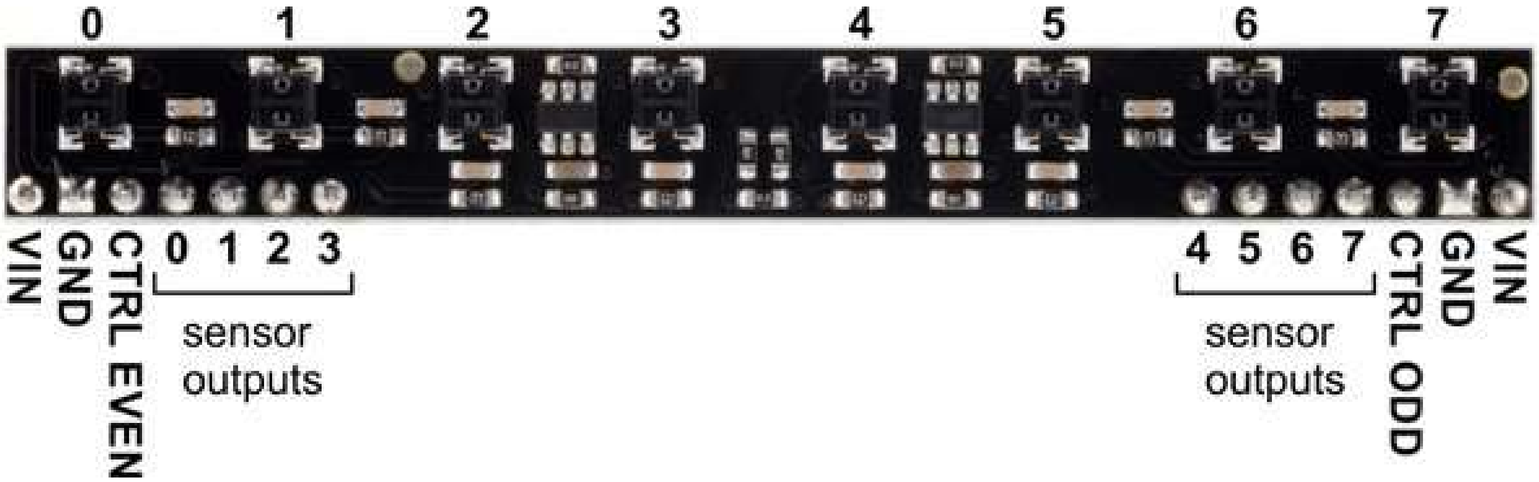
Line Tracer 04

- IR Sensor –

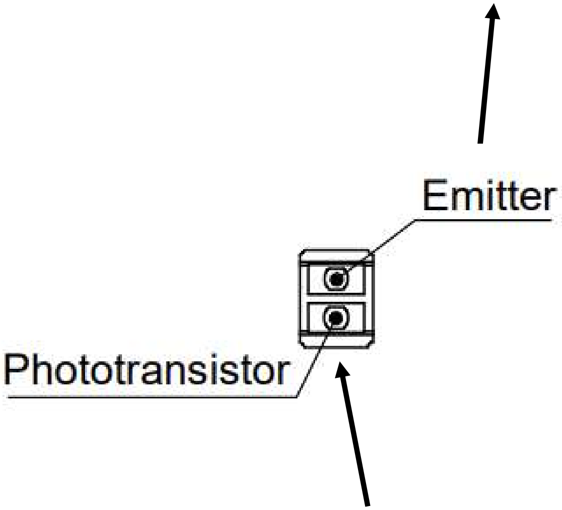
1. QTRX Sensor

About QTRX Sensor

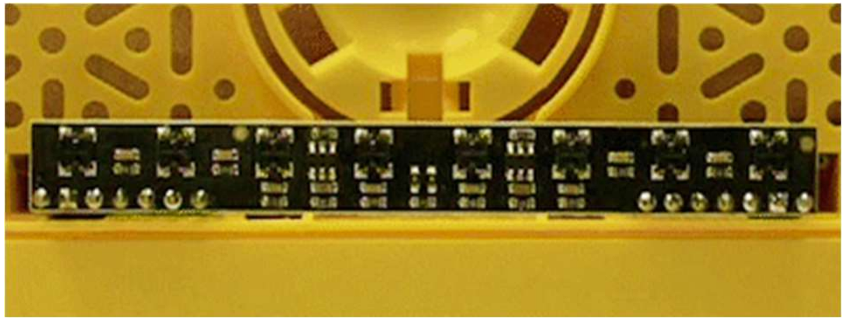


About QTRX Sensor

Infra Red light comes from here

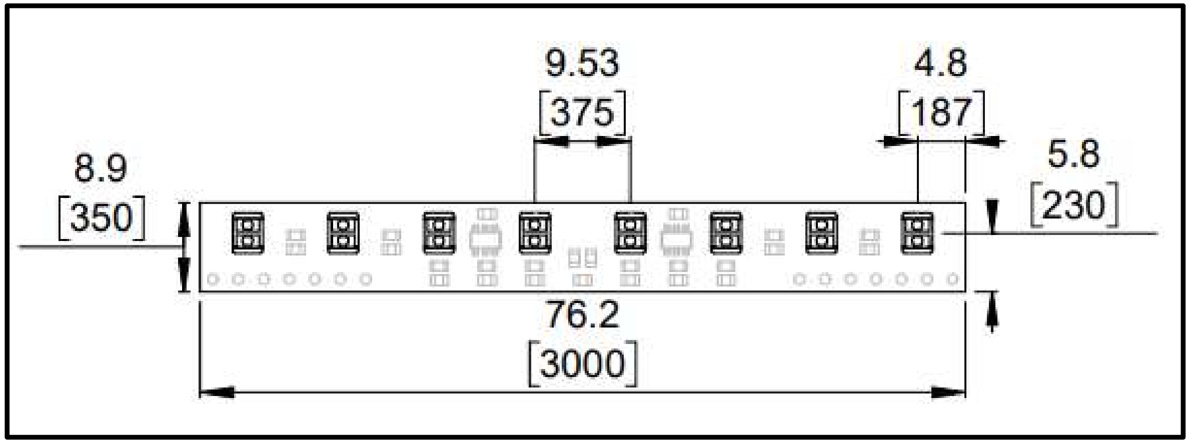


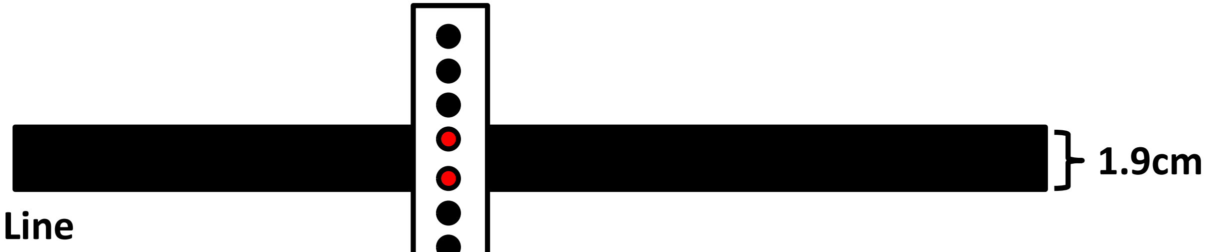
IR light should go to phototransistor



View QRTX Sensor with IR Camera

About QTRX Sensor

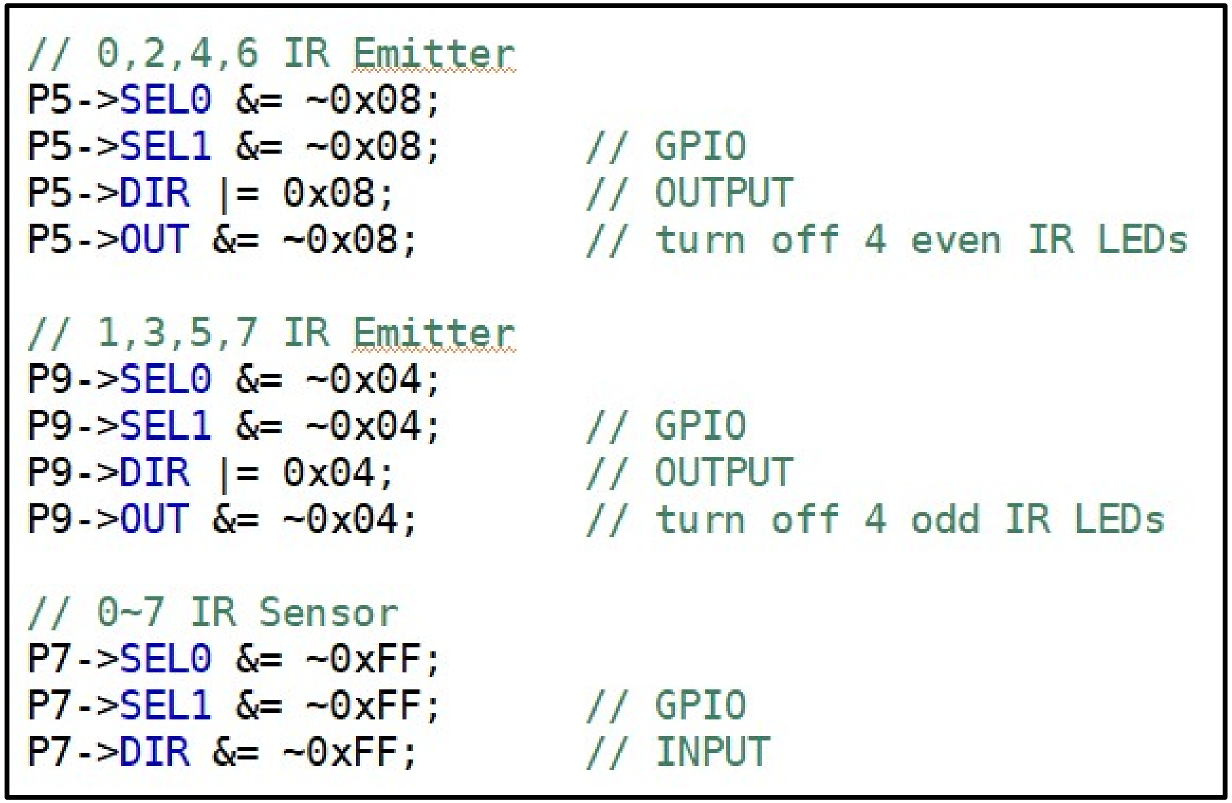




QTRX Sensor

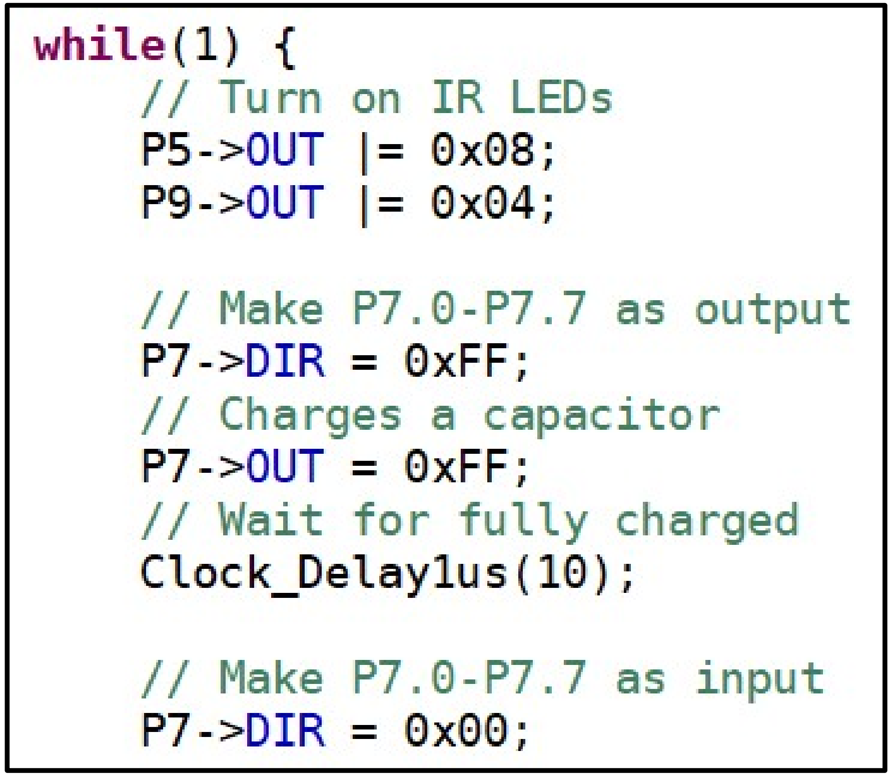
2. IR Sensor Implementation

IR Sensor Initialization



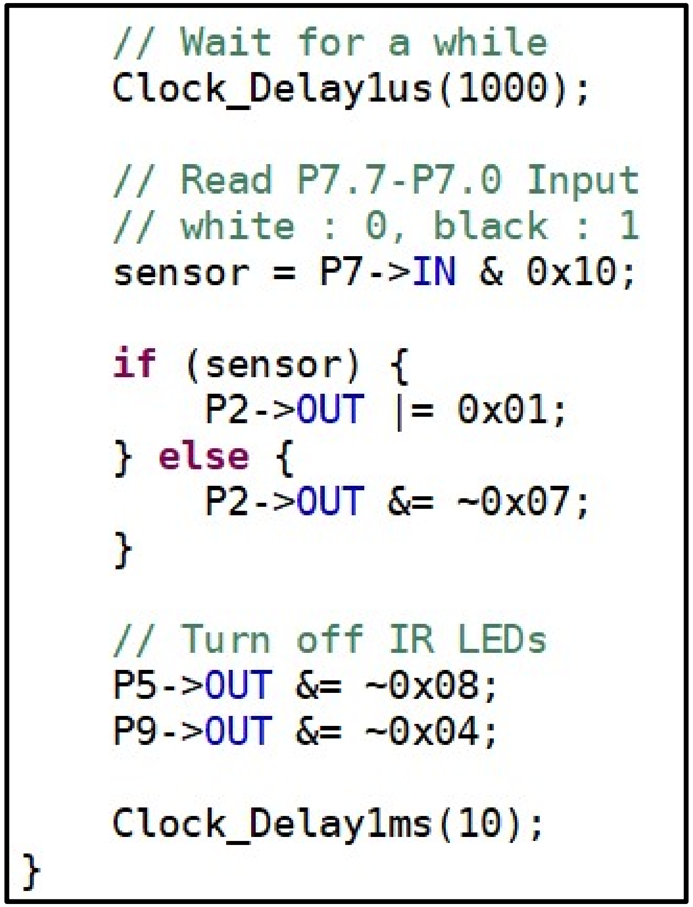
```
// 0,2,4,6 IR Emitter
P5->SEL0 &= ~0x08;
P5->SEL1 &= ~0x08;      // GPIO
P5->DIR |= 0x08;        // OUTPUT
P5->OUT &= ~0x08;       // turn off 4 even IR LEDs

// 1,3,5,7 IR Emitter
P9->SEL0 &= ~0x04;
P9->SEL1 &= ~0x04;      // GPIO
P9->DIR |= 0x04;        // OUTPUT
P9->OUT &= ~0x04;       // turn off 4 odd IR LEDs

// 0~7 IR Sensor
P7->SEL0 &= ~0xFF;
P7->SEL1 &= ~0xFF;      // GPIO
P7->DIR &= ~0xFF;       // INPUT
```

IR Sensor Basic Usage



```
while(1) {
    // Turn on IR LEDs
    P5->OUT |= 0x08;
    P9->OUT |= 0x04;

    // Make P7.0-P7.7 as output
    P7->DIR = 0xFF;
    // Charges a capacitor
    P7->OUT = 0xFF;
    // Wait for fully charged
    Clock_Delay1us(10);

    // Make P7.0-P7.7 as input
    P7->DIR = 0x00;
```

You should turn on the power!

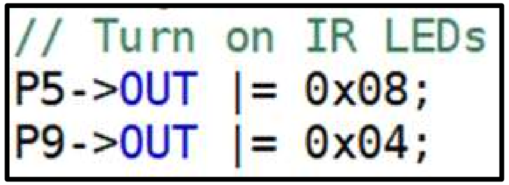


```
    // Wait for a while
    Clock_Delay1us(1000);

    // Read P7.7-P7.0 Input
    // white : 0, black : 1
    sensor = P7->IN & 0x10;

    if (sensor) {
        P2->OUT |= 0x01;
    } else {
        P2->OUT &= ~0x07;
    }

    // Turn off IR LEDs
    P5->OUT &= ~0x08;
    P9->OUT &= ~0x04;

    Clock_Delay1ms(10);
}
```

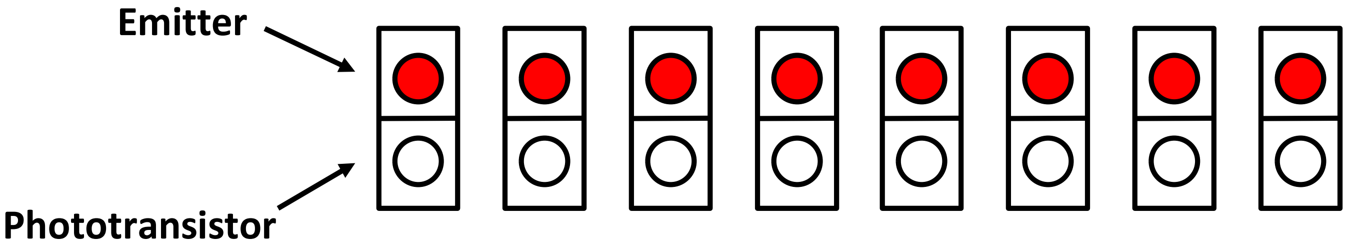
IR Sensor Basic Usage

1) Turn on IR LED

- Turn on both even and odd emitters



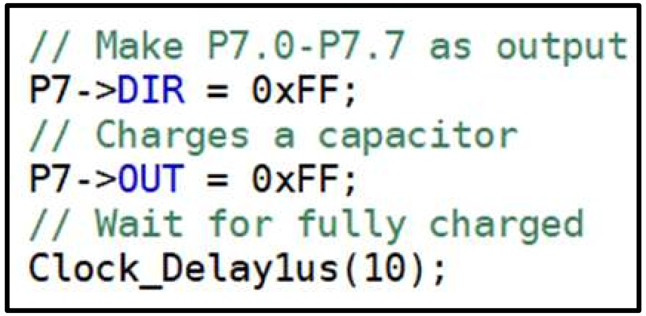
```
// Turn on IR LEDs
P5->OUT |= 0x08;
P9->OUT |= 0x04;
```



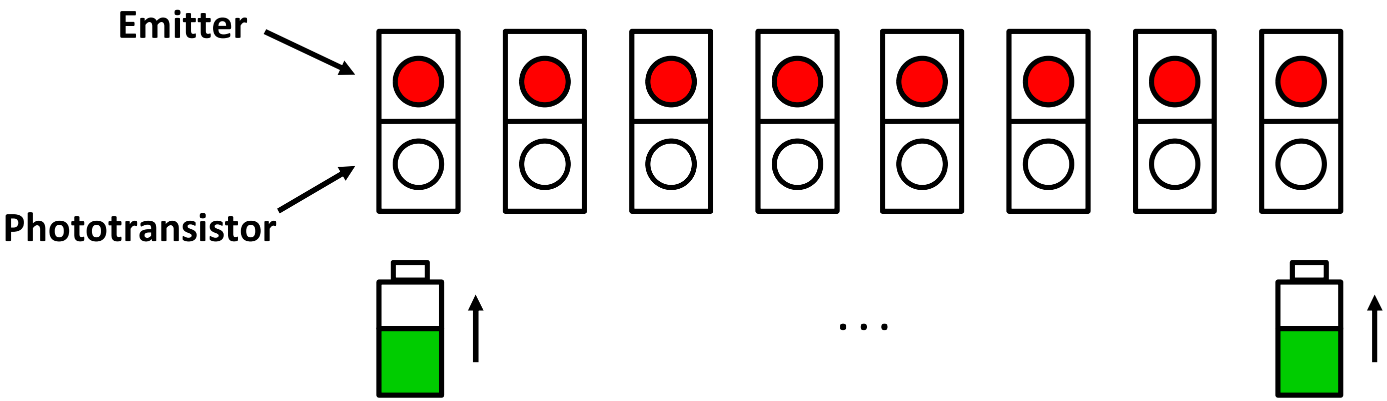
IR Sensor Basic Usage

2) Charge Capacitors

- To charge, we should change P7->DIR to output and charge capacitors through P7->OUT = 0xFF
- We need to wait for fully charged



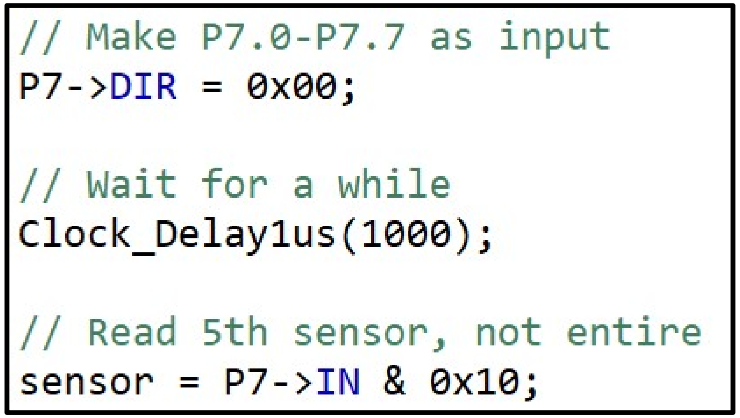
```
// Make P7.0-P7.7 as output
P7->DIR = 0xFF;
// Charges a capacitor
P7->OUT = 0xFF;
// Wait for fully charged
Clock_Delay1us(10);
```



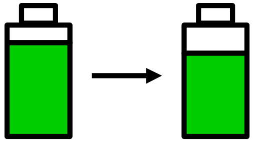
IR Sensor Basic Usage

3) Wait for a while after fully charged

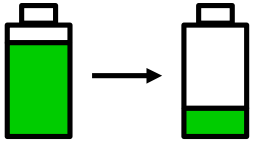
- Capacitor is discharged slowly in a natural situation, But it is very slow
- When IR Sensor gets IR light, it discharges capacitor
- Using above property, we can distinguish between white and black surfaces



```
// Make P7.0-P7.7 as input
P7->DIR = 0x00;

// Wait for a while
Clock_Delay1us(1000);

// Read 5th sensor, not entire
sensor = P7->IN & 0x10;
```



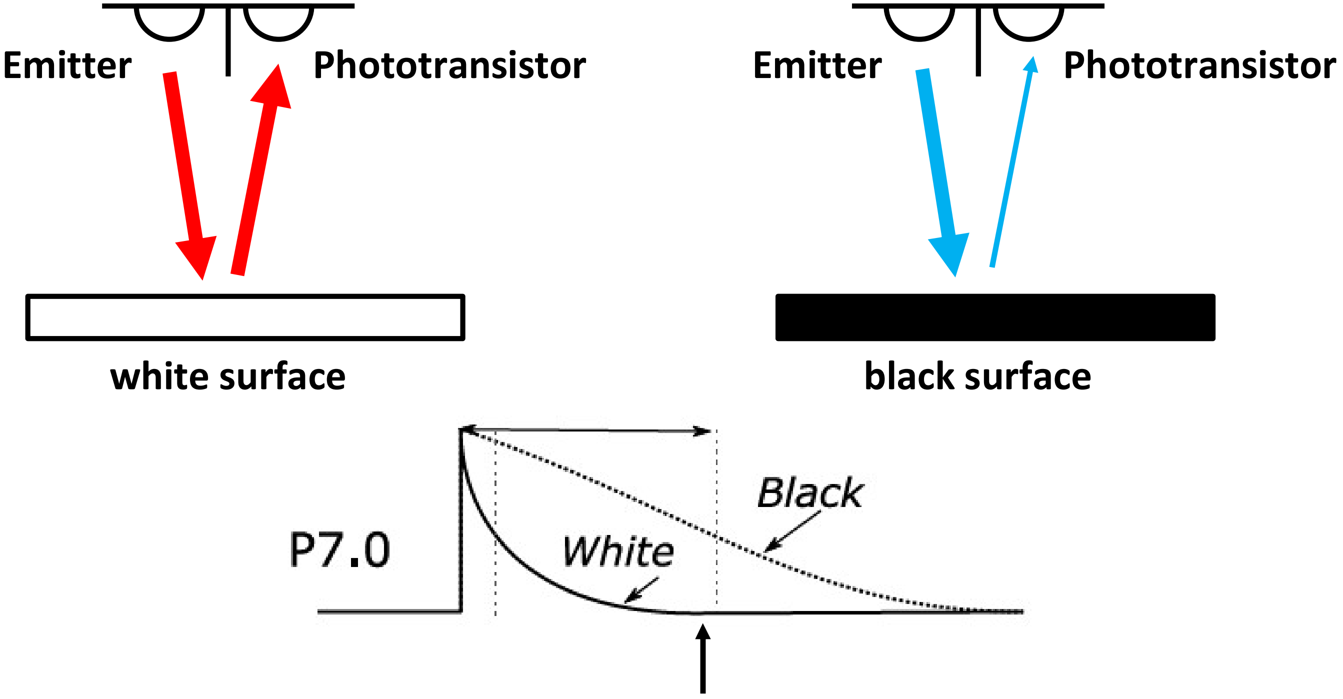
No IR Light



With IR Light

IR Sensor Basic Usage

3) Wait for a while after fully charged



Emitter

Phototransistor

white surface

Emitter

Phototransistor

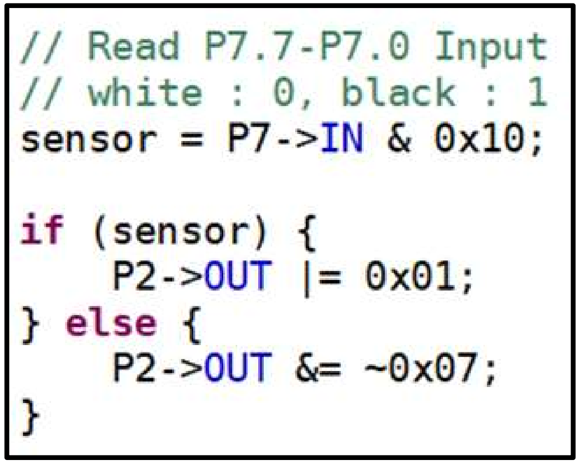
black surface

We have to read a sensor at this moment

IR Sensor Basic Usage

4) Read Sensor

- Make Port7 as input and read Port 7
- When we read 0, it means white
- When we read 1, it means black

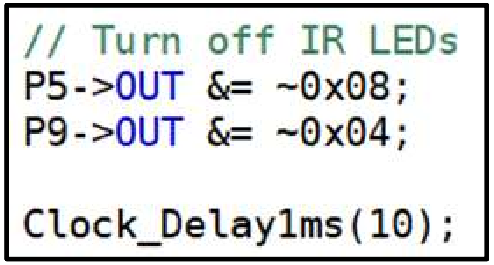


```
// Read P7.7-P7.0 Input
// white : 0, black : 1
sensor = P7->IN & 0x10;

if (sensor) {
    P2->OUT |= 0x01;
} else {
    P2->OUT &= ~0x07;
}
```

IR Sensor Basic Usage

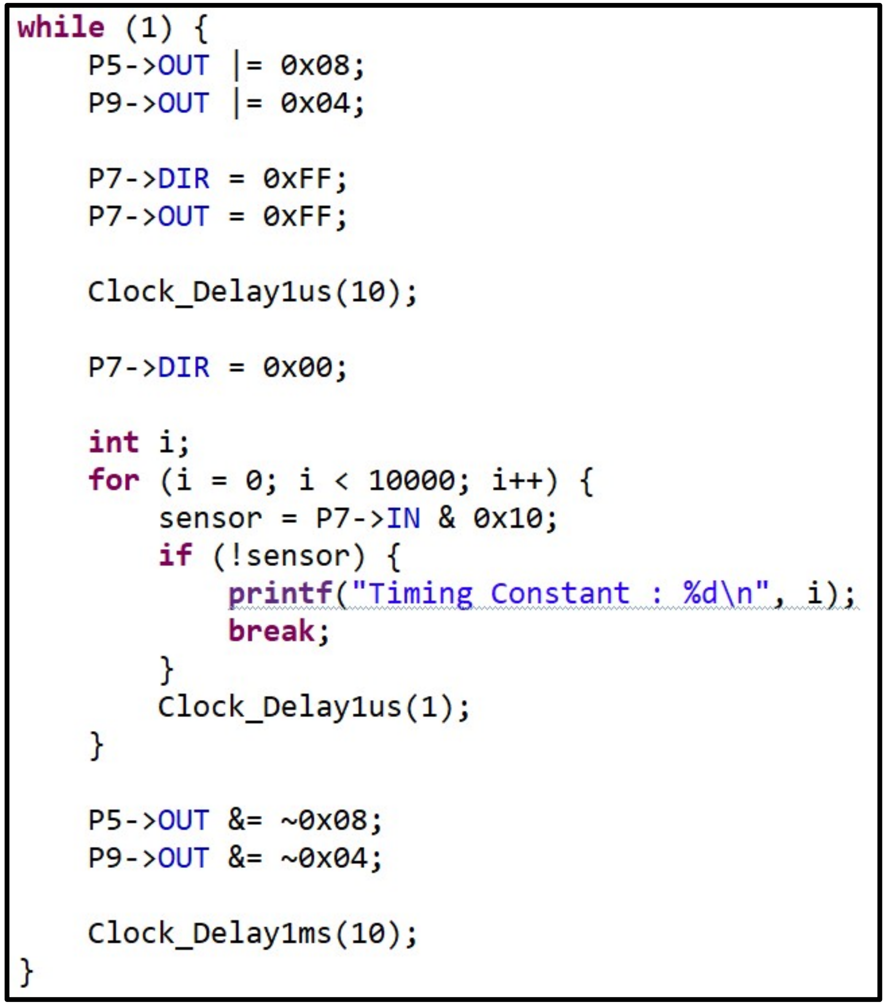
5) Turn Off LEDs

- To save energy, turn off IR LEDs and sleep for a while



```
// Turn off IR LEDs
P5->OUT &= ~0x08;
P9->OUT &= ~0x04;

Clock_Delay1ms(10);
```

Tip for Setting Waiting Constant



```
while (1) {
    P5->OUT |= 0x08;
    P9->OUT |= 0x04;

    P7->DIR = 0xFF;
    P7->OUT = 0xFF;

    Clock_Delay1us(10);

    P7->DIR = 0x00;

    int i;
    for (i = 0; i < 10000; i++) {
        sensor = P7->IN & 0x10;
        if (!sensor) {
            printf("Timing Constant : %d\n", i);
            break;
        }
        Clock_Delay1us(1);
    }

    P5->OUT &= ~0x08;
    P9->OUT &= ~0x04;

    Clock_Delay1ms(10);
}
```

```
Timing Constant : 1713
Timing Constant : 1671
Timing Constant : 1689
...
```

No Reflection

```
Timing Constant : 311
Timing Constant : 305
Timing Constant : 310
...
```

White Surface

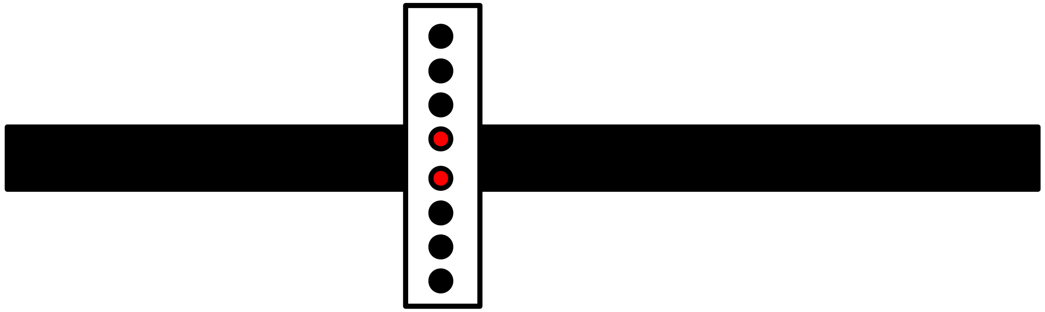
```
Timing Constant : 790
Timing Constant : 785
Timing Constant : 791
```

Black Surface

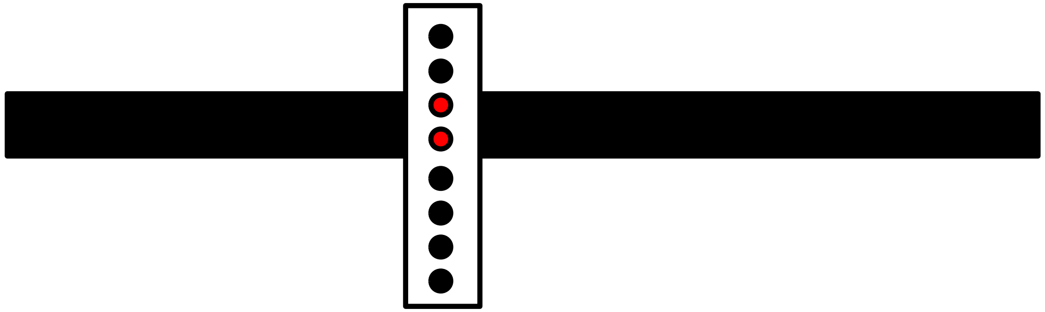
3. IR Sensor Activity

Line Follower - Sensor

Turn on LED when the line is located at the center of the robot



Turn on LED!



Turn off LED!

- PWM & DC Motor -

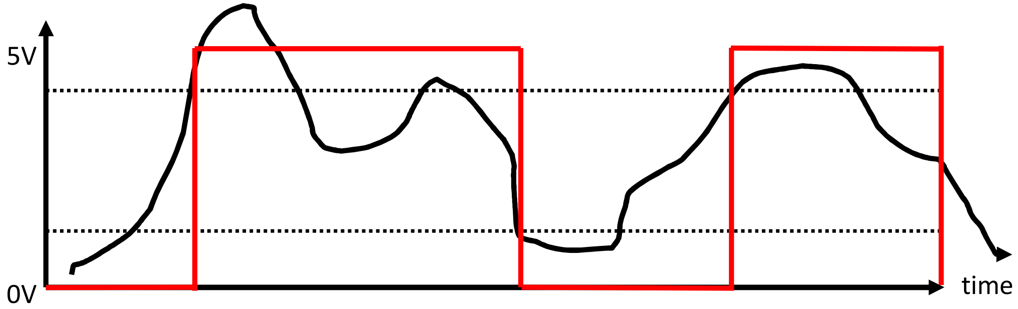
1. PWM

PWM Principle

We want to adjust the brightness of the LED

- 0V means 0% brightness
- 5V means 100% brightness
- 0.05V means 1% brightness?

-> No. Circuit would consider 0.05V as 0V
We need a way to convert a digital signal into an analog signal



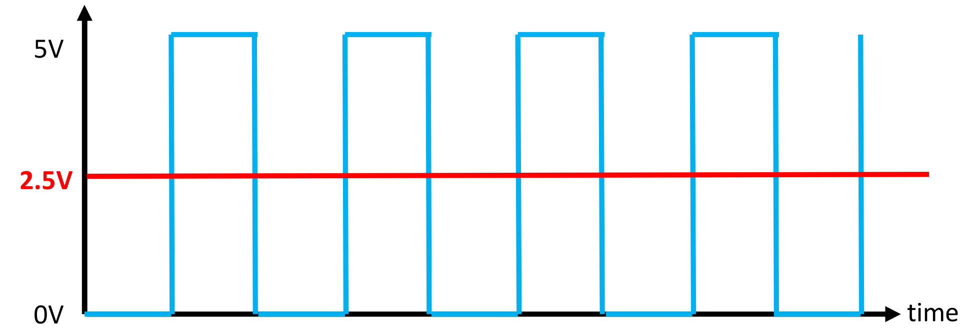
PWM Principle

What is PWM

- Pulse Width Modulation
- Digital to Analog Converter

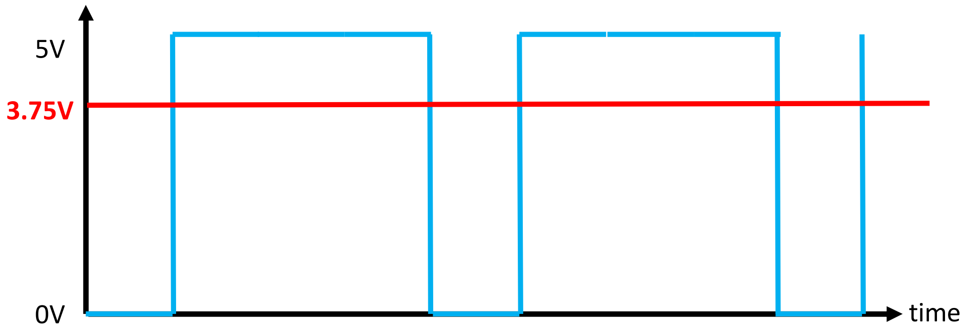
PWM Principle

50% duty cycle



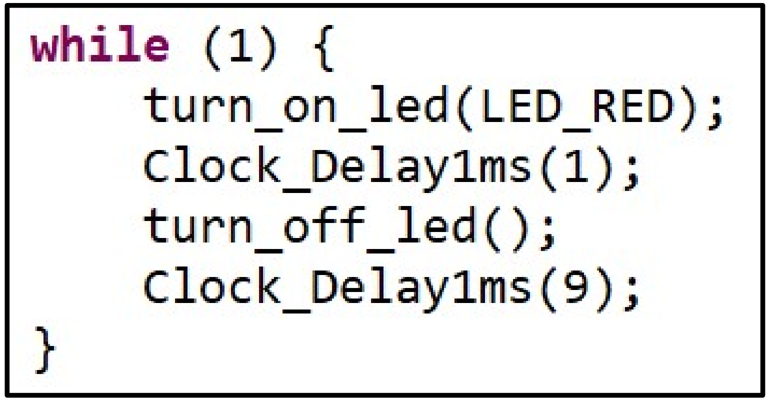
It behaves like 2.5v

75% duty cycle



It behaves like 3.75v

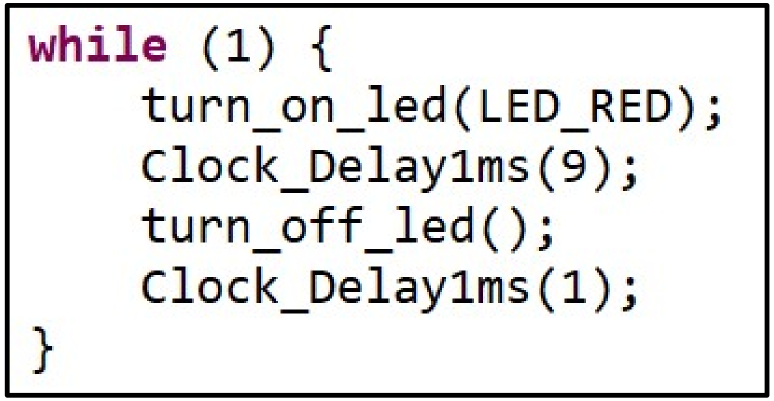
PWM Example 1



```
while (1) {
    turn_on_led(LED_RED);
    Clock_Delay1ms(1);
    turn_off_led();
    Clock_Delay1ms(9);
}
```

PWM Freq : 100Hz

Duty Cycle : 10%

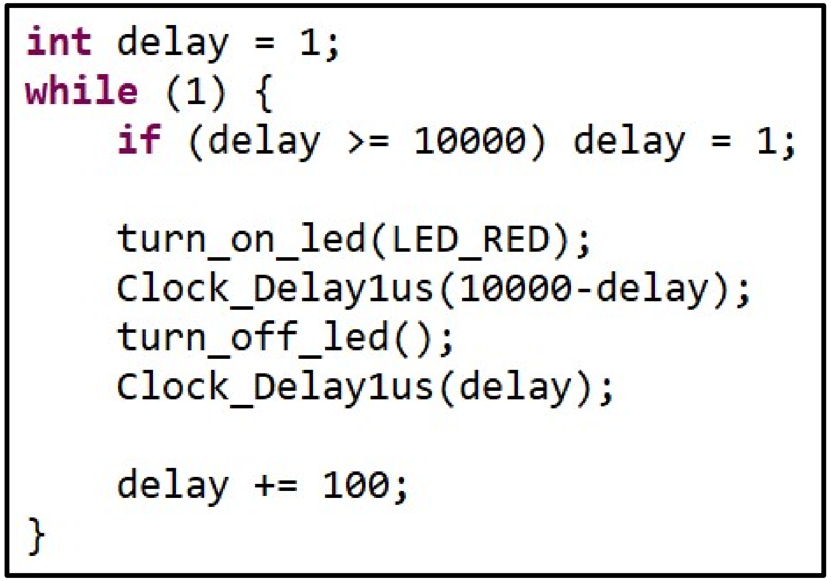


```
while (1) {
    turn_on_led(LED_RED);
    Clock_Delay1ms(9);
    turn_off_led();
    Clock_Delay1ms(1);
}
```

PWM Freq : 100Hz

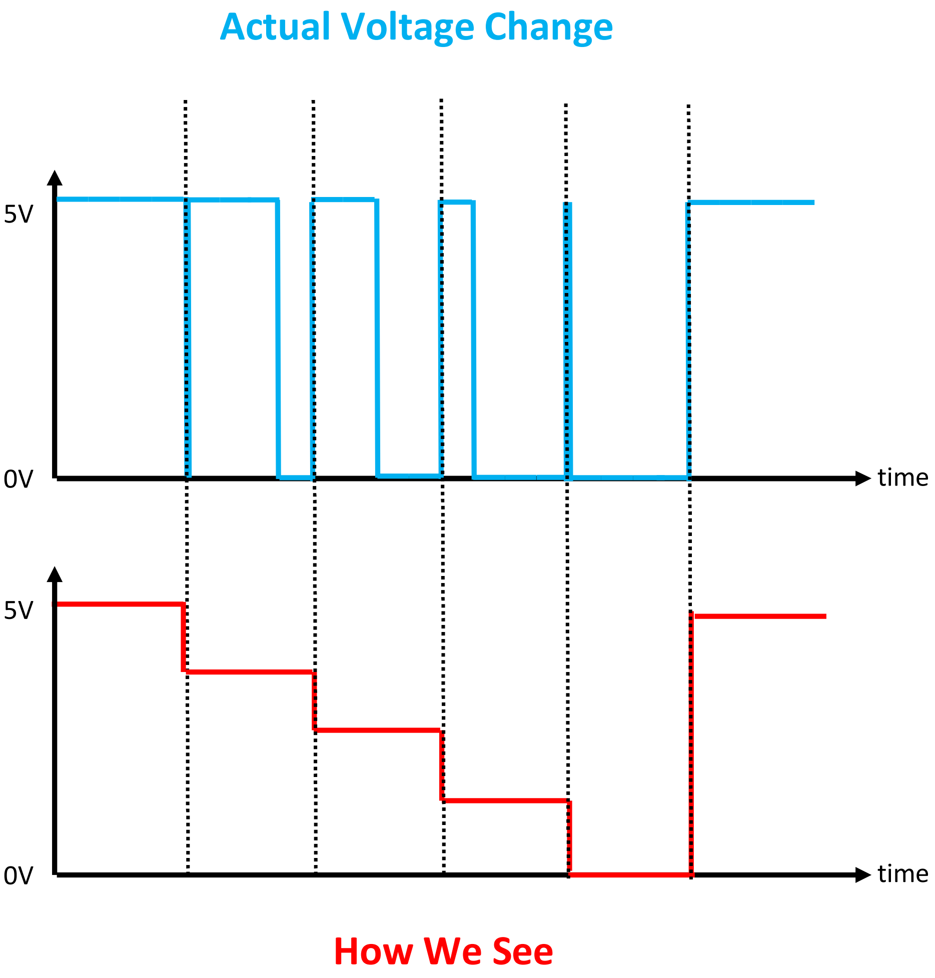
Duty Cycle : 90%

PWM Example 2



```
int delay = 1;
while (1) {
    if (delay >= 10000) delay = 1;

    turn_on_led(LED_RED);
    Clock_Delay1us(10000-delay);
    turn_off_led();
    Clock_Delay1us(delay);

    delay += 100;
}
```

100% brightness -> 0% brightness for every second



2. Motor

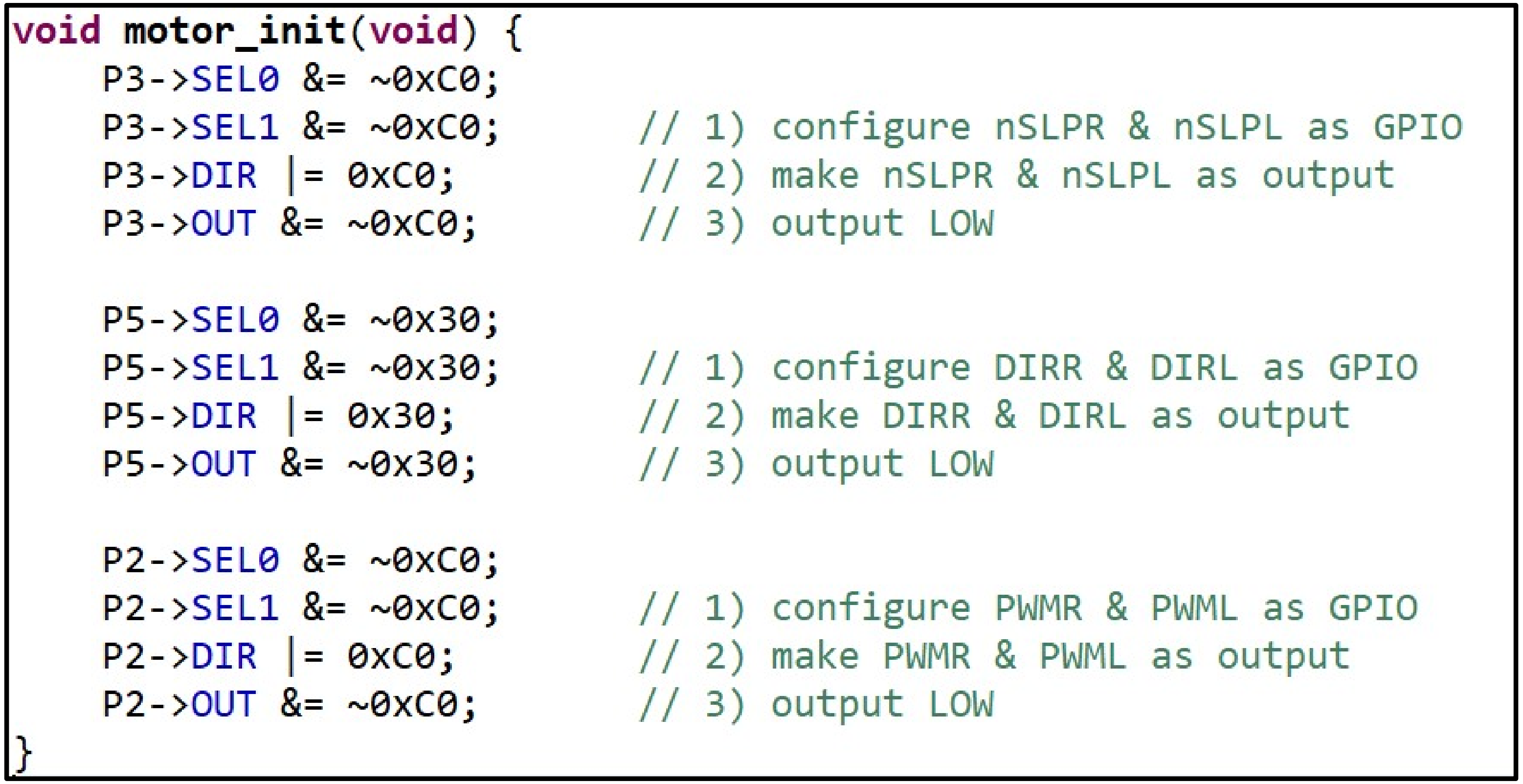
Motor Port Map

LaunchPad	TI-RSLK chassis board	DRV8838	Description
P5.5	DIRR	PH	Right Motor Direction
P3.6	nSLPR	nSLEEP	Right Motor Sleep
P2.6	PWMR	EN	Right Motor PWM
P5.4	DIRL	PH	Left Motor Direction
P3.7	nSLPL	nSLEEP	Left Motor Sleep
P2.7	PWML	EN	Left Motor PWM

PH	EN	nSleep	Motor
0	0	1	Stop
1	0	1	Stop
0	1	1	Forward
1	1	1	Back

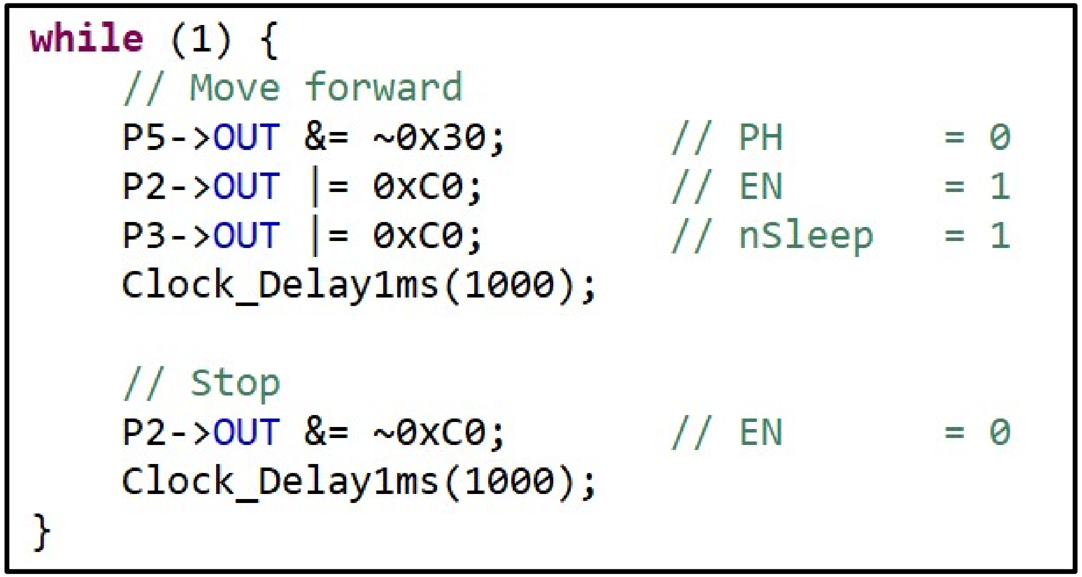
To go forward, set nSleep=1, PH=0, and activate EN

Motor Initialization



```
void motor_init(void) {
    P3->SEL0 &= ~0xC0;
    P3->SEL1 &= ~0xC0;      // 1) configure nSLPR & nSLPL as GPIO
    P3->DIR |= 0xC0;        // 2) make nSLPR & nSLPL as output
    P3->OUT &= ~0xC0;       // 3) output LOW

    P5->SEL0 &= ~0x30;
    P5->SEL1 &= ~0x30;      // 1) configure DIRR & DIRL as GPIO
    P5->DIR |= 0x30;        // 2) make DIRR & DIRL as output
    P5->OUT &= ~0x30;       // 3) output LOW

    P2->SEL0 &= ~0xC0;
    P2->SEL1 &= ~0xC0;      // 1) configure PWMR & PWML as GPIO
    P2->DIR |= 0xC0;        // 2) make PWMR & PWML as output
    P2->OUT &= ~0xC0;       // 3) output LOW
}
```

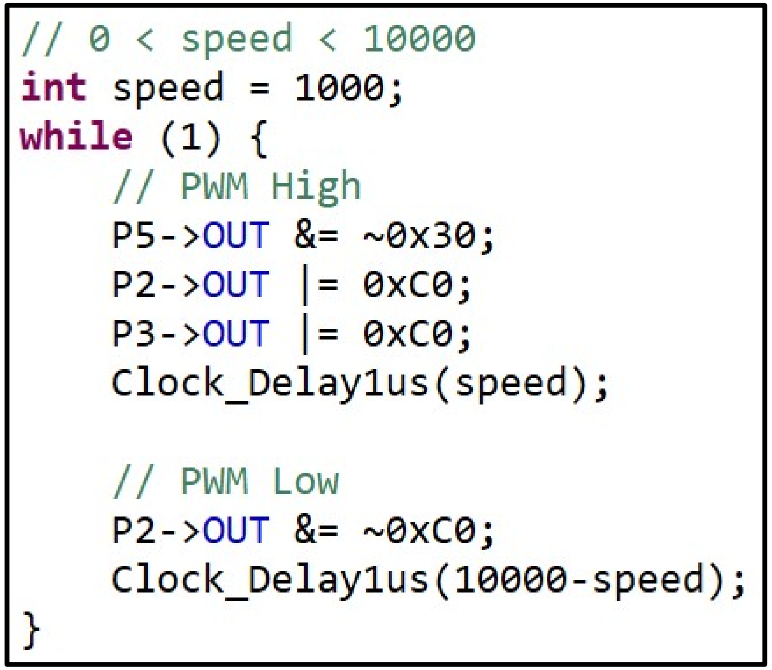
Motor Example



```
while (1) {
    // Move forward
    P5->OUT &= ~0x30;        // PH       = 0
    P2->OUT |= 0xC0;         // EN       = 1
    P3->OUT |= 0xC0;         // nSleep   = 1
    Clock_Delay1ms(1000);

    // Stop
    P2->OUT &= ~0xC0;        // EN       = 0
    Clock_Delay1ms(1000);
}
```

You should turn on the power!

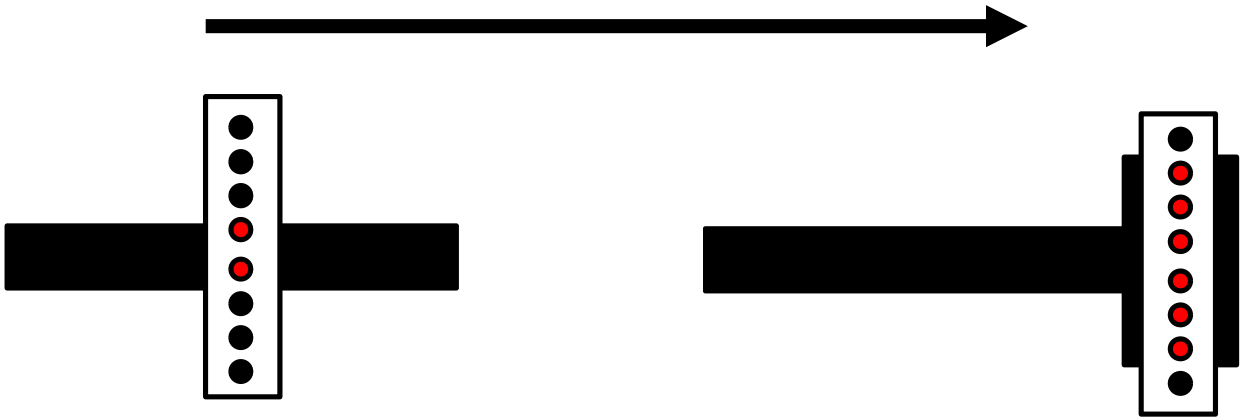
Motor Speed Control Example



```
// 0 < speed < 10000
int speed = 1000;
while (1) {
    // PWM High
    P5->OUT &= ~0x30;
    P2->OUT |= 0xC0;
    P3->OUT |= 0xC0;
    Clock_Delay1us(speed);

    // PWM Low
    P2->OUT &= ~0xC0;
    Clock_Delay1us(10000-speed);
}
```

3. Motor Activity

Stop at finish line



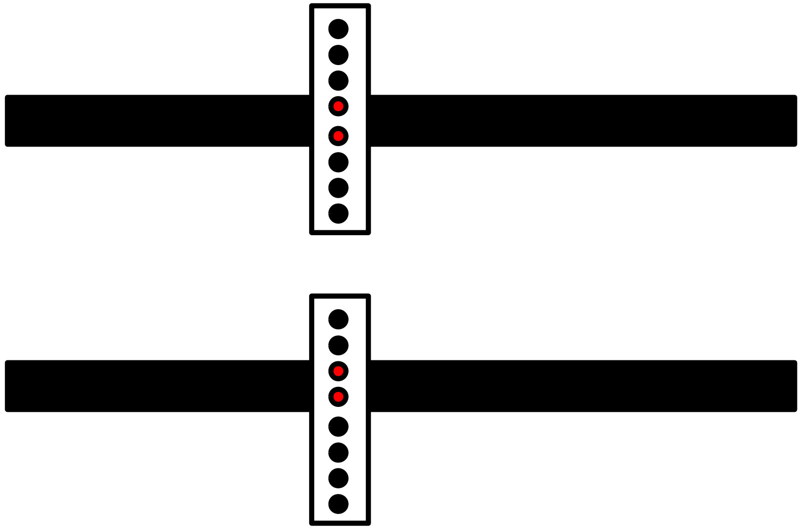
Move!

Don’t Move!

4. Assignment

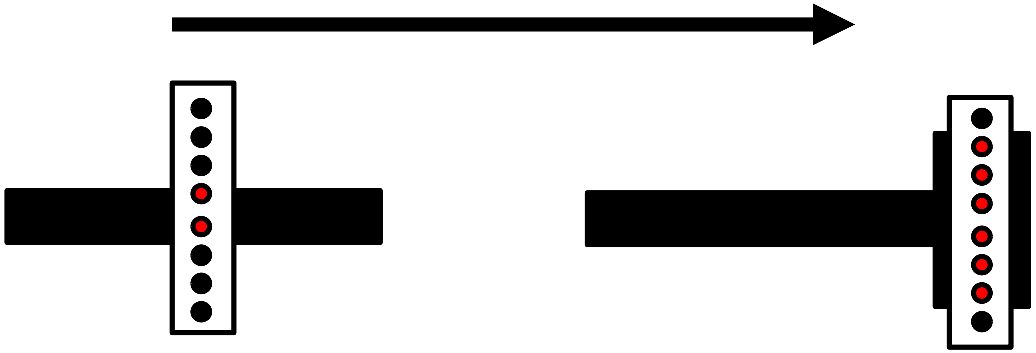
Submission guide (~10/18)

Turn on LED when the line is located at the center of the robot



Turn on LED!

Turn off LED!



Move!

Don't Move!