Smart Software Project

Lecture: Week 11

Line Tracing & Accessing Location Information

Prof. HyungJune Lee
hyungjune.lee@ewha.ac.kr

이화여자대학교
EWHA WOMANS UNIVERSITY

Today

- Review from the last lecture
  - Infrared sensors

- Line tracing
- Accessing location information from Android

- Announcement

Class Schedule

Week	Lecture Contents	Lab Contents
Week 1	Course introduction	Arduino introduction: platform & programming environment
Week 2	Embedded system overview & source management in collaborative repository (using GitHub)	Lab 1: Arduino Mega 2560 board & SmartCAR platform
Week 3	ATmega2560 Micro-controller (MCU): architecture & I/O ports, Analog vs. Digital, Pulse Width Modulation	Lab 2: SmartCAR LED control
Week 4	Analog vs. Digital & Pulse Width Modulation	Lab 3: SmartCAR motor control (Due: HW on creating project repository using GitHub)
Week 5	ATmega2560 MCU: memory, I/O ports, UART	Lab 4: SmartCAR control via Android Bluetooth
Week 6	ATmega2560 UART control & Bluetooth communication between Arduino platform and Android device	Lab 5: SmartCAR control through your own customized Android app (Due: Project proposal)
Week 7	Midterm exam	
Week 8	ATmega2560 Timer, Interrupts & Ultrasonic sensors	Lab 6: SmartCAR ultrasonic sensing
Week 9	Infrared sensors & Buzzer	Lab 7: SmartCAR infrared sensing
Week 10	Acquiring location information from Android device & line tracing	Lab 8: Implementation of line tracer
Week 11	Gyroscope, accelerometer, and compass sensors	Lab 9: Using gyroscope, accelerometer, and compass sensors
Week 12	Project	Team meeting (for progress check)
Week 13	Project	Team meeting (for progress check)
Week 14	Course wrap-up & next steps	
Week 15	Project presentation & demo I (Due: source code, presentation slides, & poster slide)	Project presentation & demo II
Week 16	Final week (no final exam)	

Today

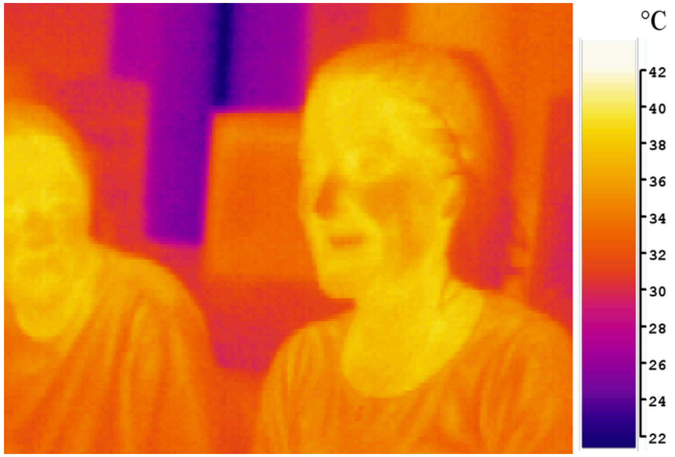
- **Review from the last lecture**
  - **Infrared sensors**

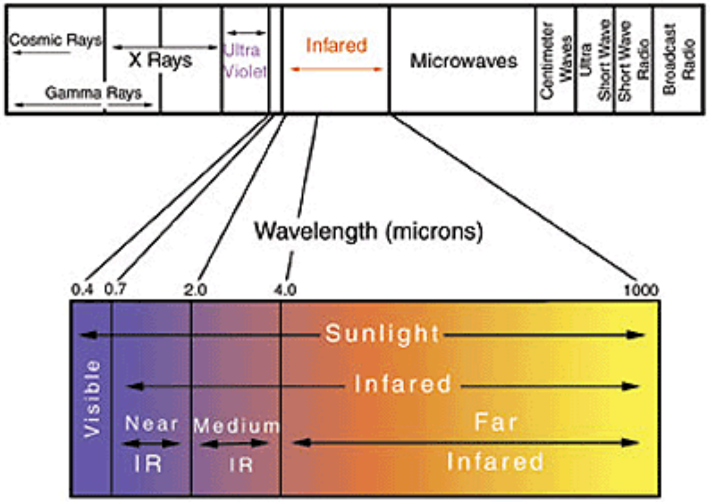
- Line tracing
- Accessing location information from Android

- Announcement

Infrared Light

- Infrared light
  - Electromagnetic radiation with longer wavelengths
  - Wavelength: 0.75 μm ~ 1 mm
  - Beyond red light in light spectrum
  - Most of thermal radiation emitted by objects near room temperature is infrared
    - ~ Few μm: near IR
    - > 25μm : far IR
    - In between: medium IR

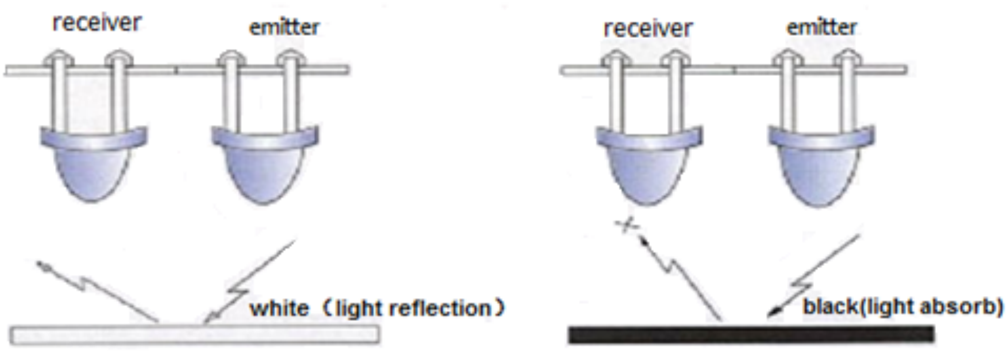




Infrared Sensors in SmartCAR

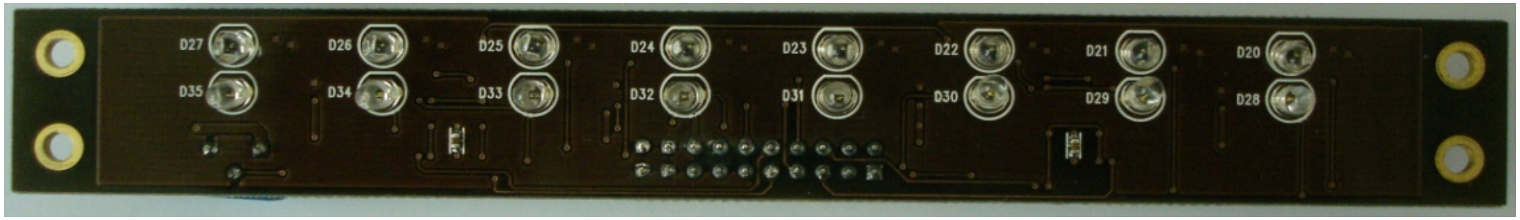
- Infrared sensors in SmartCAR
  - Emitter
    - Infrared Diode(EL-8L): electrical signal to infrared light
  - Receiver
    - Phototransistor(ST-8L): infrared light to electrical signal

  - 1) Infrared light transmitted at Infrared Diode is reflected from the surrounding object
    2) The reflected light is detected at Phototransistor(ST-8L)
  - The amount of detected light at receiver varies depending on the darkness level of the reflected surface
  - Functionality
    - Detect line status in the bottom using 8 sets of infrared sensors
    - Based on these inputs, motors will be controlled



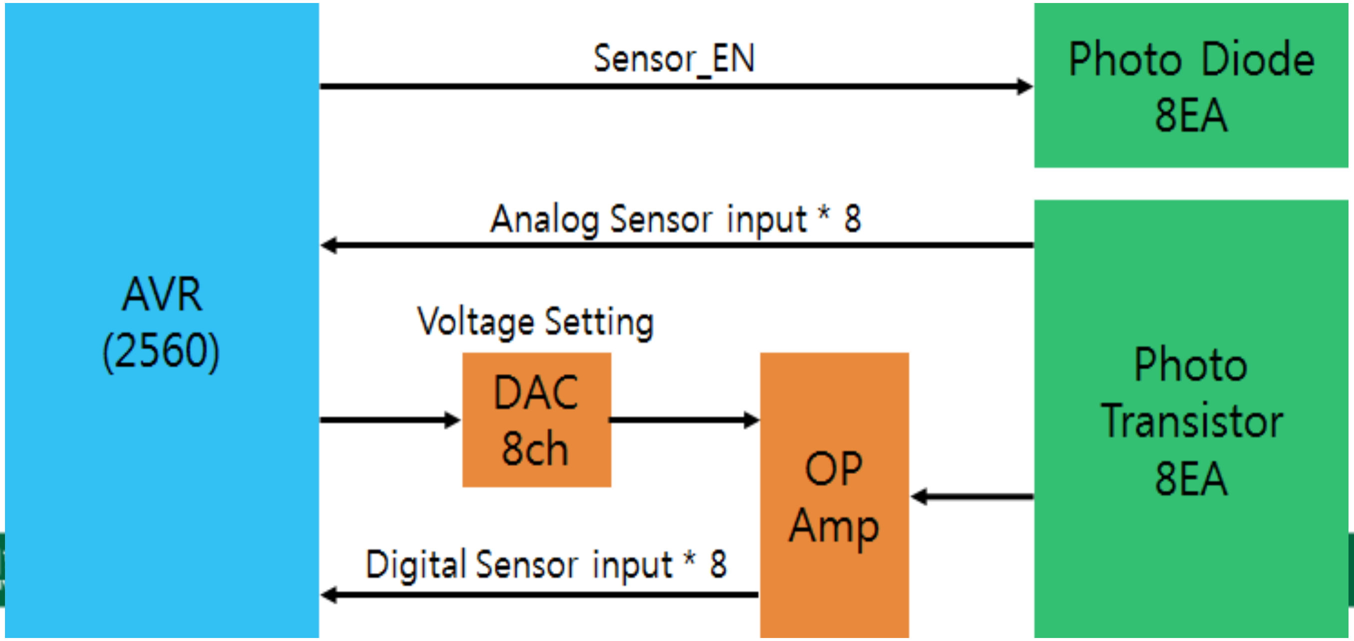
Infrared Sensors in SmartCAR

- Infrared sensors in SmartCAR
  - Functionality
    - Detect line status in the bottom using 8 sets of infrared sensors
    - Based on these inputs, motors will be controlled



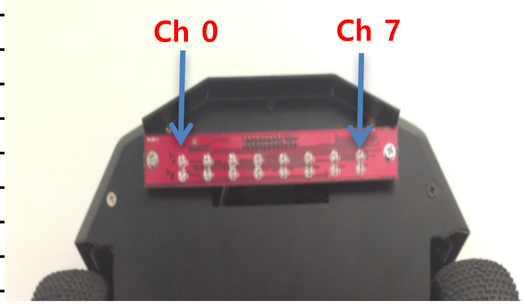
Infrared Sensors in SmartCAR

- Sensor_EN
  - Enable infrared sensors
- Analog sensor input
  - Measure infrared level in analog
- Digital sensor input
  - measure infrared level in digital
- Block diagram of infrared sensors in SmartCAR



SmartCAR Infrared Sensor Port Configuration

Type	Name	Port / Number	Etc
Digital Input	SENSOR_1 (LEFTMOST)	PC7 / 30	
	SENSOR_2	PC6 / 31	
	SENSOR_3	PC5 / 32	
	SENSOR_4	PC4 / 33	
	SENSOR_5	PC3 / 34	
	SENSOR_6	PC2 / 35	
	SENSOR_7	PC1 / 36	
	SENSOR_8 (RIGHTMOST)	PC0 / 37	
Analog Input	SENA_1 (LEFTMOST)	PF0 / A0	
	SENA_2	PF1 / A1	
	SENA_3	PF2 / A2	
	SENA_4	PF3 / A3	
	SENA_5	PF4 / A4	
	SENA_6	PF5 / A5	
	SENA_7	PF6 / A6	
	SENA_8 (RIGHTMOST)	PF7 / A7	
	SEN_EN	PA4 / 26	
DAC	S_DIN	PL7 / 42	
	S_SCLK	PL6 / 43	
	S_SYNCN	PL5 / 44	



- 30~37: ports to read digital values at receiver based on reference voltage set-up in OP AMP
- A0~A7: ports to read analog values at receiver
- 26: enable infrared emitter - ‘1’ turning on emitter
- 42~44: ports for configuring reference voltage in Serial DAC
  - Configure reference voltages for 8 pins in OP AMP

Serial DAC Control

- Serial DAC Data Format (16-bit integer)

0	CH2	CH1	CH0	D7	D6	D5	D4	D3	D2	D1	D0	X	X	X	X

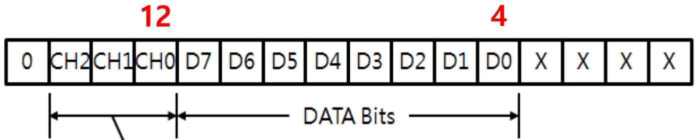
DATA Bits

CH2	CH1	CH0	channel
0	0	0	A
0	0	1	B
0	1	0	C
0	1	1	D
1	0	0	E
1	0	1	F
1	1	0	G
1	1	1	H

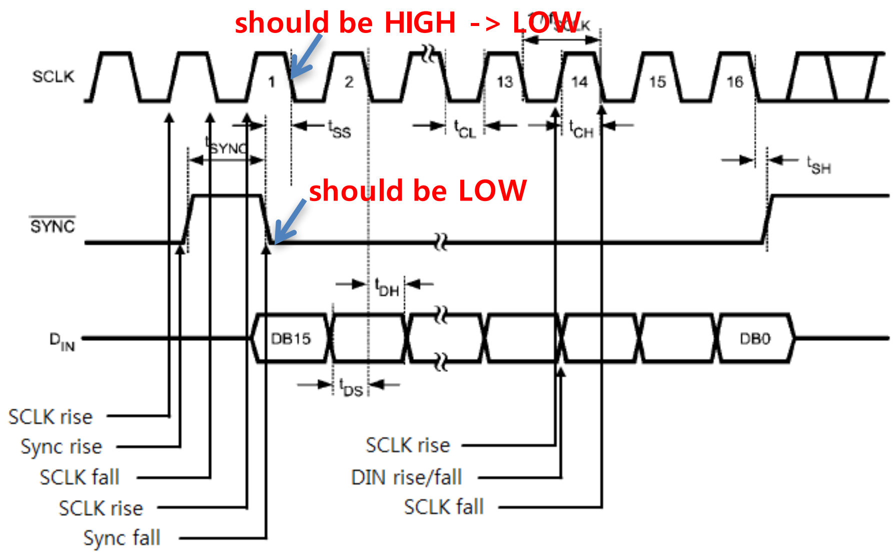
  - CH2, CH1, CH0: channel data to select one among A ~ H
  - DAC data bits should be sent one-by-one from MSB (Most Significant Bit) first
  - Last 4 bits: garbage data

Serial DAC Control

```
void DAC_CH_Write(unsigned int ch, unsigned int da)
{
    unsigned int data = ((ch << 12) & 0x7000) |
                        ((da << 4) & 0x0FF0);
    DAC_setting(data);
}

void DAC_setting(unsigned int data)
{
   int z;

   digitalWrite(S_SCLK,HIGH);
   delayMicroseconds(1);
   digitalWrite(S_SCLK,LOW);
   delayMicroseconds(1);
   digitalWrite(S_SYNCN,LOW);
   delayMicroseconds(1);
   for(z=15;z>=0;z--)
   {
     digitalWrite(S_DIN,(data>>z)&0x1);
     digitalWrite(S_SCLK,HIGH);
     delayMicroseconds(1);
     digitalWrite(S_SCLK,LOW);
     delayMicroseconds(1);
   }
   digitalWrite(S_SYNCN,HIGH);
}
```



CH2	CH1	CH0	channel
0	0	0	A
0	0	1	B
0	1	0	C
0	1	1	D
1	0	0	E
1	0	1	F
1	1	0	G
1	1	1	H



Threshold for Digital Sensor Input Decision

- To divide between dark area and bright area based on a threshold

- Experiment on the analog value on "white"
- Experiment on the analog value on "black"
- Set the average value to DAC

Threshold for Digital Sensor Input Decision

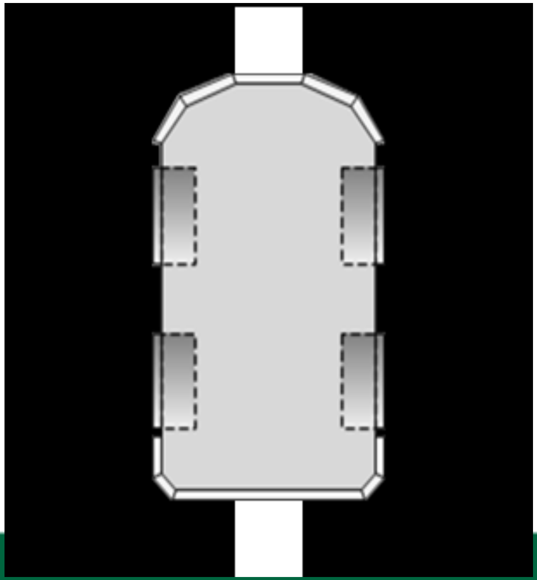
```
#define S_DIN          42
#define S_SCLK         43
#define S_SYNCN        44
#define IN_SEN_EN      26

int SensorA[8] = {A0,A1,A2,A3,A4,A5,A6,A7};
int SensorD[8] = {30,31,32,33,34,35,36,37};

void setup()
{
    int z;
    int dac_val_min[8] =
                    {59,94,81,79,166,104,108,77};
    int dac_val_max[8] =
                {443,627,678,603,957,761,797,559};

    Serial.begin(115200);

    pinMode(IN_SEN_EN,OUTPUT);
    pinMode(S_DIN,OUTPUT);
    pinMode(S_SCLK,OUTPUT);
    pinMode(S_SYNCN,OUTPUT);
    digitalWrite(S_SCLK,LOW);
    digitalWrite(S_SYNCN,HIGH);
    digitalWrite(IN_SEN_EN,HIGH);
```

Mode	DB[15:12]	DB[11:0]	Etc
WRM	1000	XXXX XXXX XXXX	0x8000
WTM	1001	XXXX XXXX XXXX	0x9000

```
    for (z=0; z<8; z++)
        pinMode(SensorD[z], INPUT);

    DAC_setting(0x9000); //for Write-Through Mode

    for (z=0; z<8; z++)
    {
        int mean_val =
       (dac_val_min[z]+dac_val_max[z])/2;   //10-bit

        DAC_CH_Write(z, mean_val >> 2);
        //should be 8-bit
    }
}
```

Example Program

- SmartCAR is on a while track
  - If an infrared sensor is on "black", it prints out '0' to UART
  - If on "white", it prints out '1' to UART
    - For example, in the following figure, "0011 1100" or "0001 1000"
      - Left and right sides: '0'
      - Center: '1'



Print measurements to UART

```
void infrared_sensor_read()
{
    int z;

    for(z=7;z>=0;z--)
    {
        unsigned int val = analogRead(SensorA[z]);

        Serial.print(val);
        Serial.print("  ");
    }

    Serial.println("");

    for(z=7;z>=0;z--)
    {
        unsigned int val = digitalRead(SensorD[z]);
        Serial.print(val);
        Serial.print("  ");
    }
}
```

```
void serialEvent()
{
    int command = Serial.read();

    switch (command)
    {
        …
        …
        case 11:
            infrared_sensor_read();
            break;

        default:
    }
}
```

- If the SmartCAR receives a byte of 11, it prints out
  - analog values
  - digital values

  from 8 infrared sensors

Summary of Steps

- 1) Run an application to measure analog values
  - Measure Analog Infrared Sensor Value on "White"
  - Measure Analog Infrared Sensor Value on "Black"
- 2) Add the voltage setup for digital in setup()
  - Set up the average value
- 3) Run an application to measure analog values as well as digital values

Today

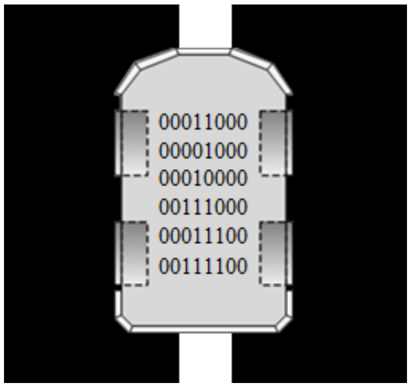
- Review from the last lecture
  - Infrared sensors

- **Line tracing**
- Accessing location information from Android

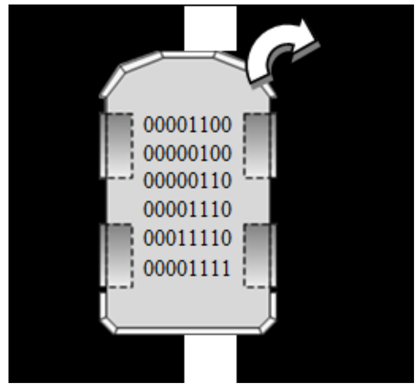
- Announcement

Line Tracer

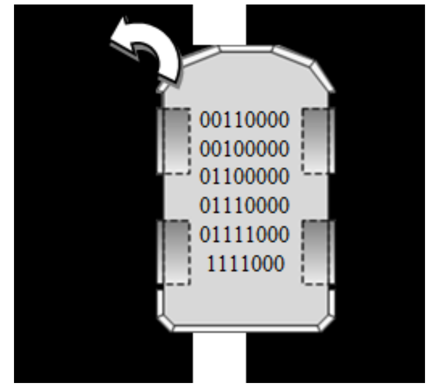
- Line tracing in SmartCAR
  - Infrared sensor data depending on SmartCAR's position



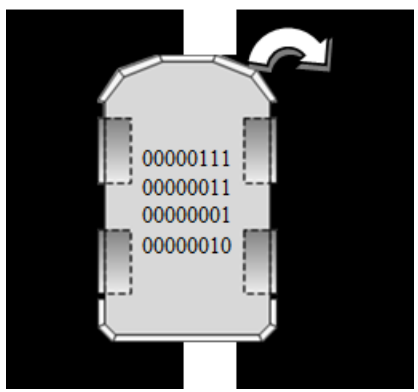
(a) Forward



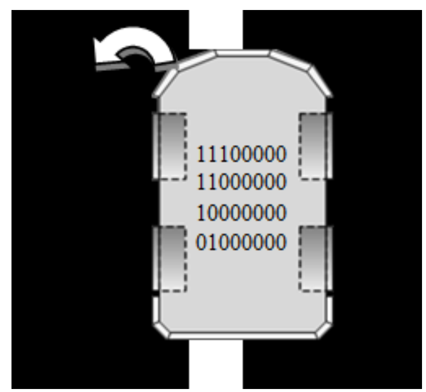
(b) Smooth Right-turn



(c) Smooth Left-turn

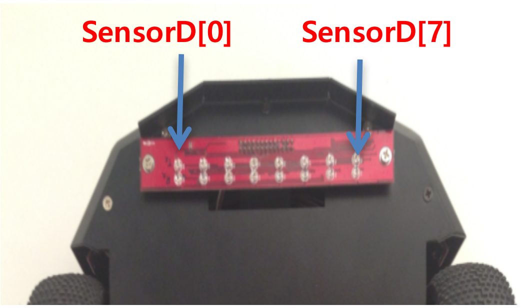


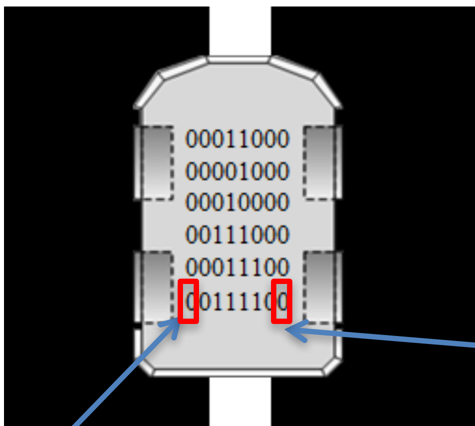
(d) Pivot Right-turn



(e) Pivot Left-turn

Sensor Data






```
…
…

unsigned char sensor_data = 0;
int z;

for(z=0;z<8;z++)
{
    unsigned int val = digitalRead(SensorD[z]);
    sensor_data |= (val << z);
}
```

SensorD[7]	SensorD[6]	SensorD[5]	SensorD[4]	SensorD[3]	SensorD[2]	SensorD[1]	SensorD[0]

To track black line in white background,
- we should complement the sensor_data ('1' to '0', '0' to '1')

```
sensor_data = ~sensor_data;
```

Control Motors w.r.t. Infrared Sensor

- How to control motors w.r.t. sensor_data

Sensor_data	Direction	Speed_data_L	Speed_data_R	Etc
0x18	FORWARD	140	140	Forward
0x10				
0x08				
0x38				
0x1C				
0x3C				
0x0C	RIGHT	200	0	Smooth Right Turn
0x04				
0x06				
0x0E				
0x1E				
0x0F				
0x30	LEFT	0	200	Smooth Left Turn
0x20				
0x60				
0x70				
0x78				
0xF0				
0x07	PIVOT_RIGHT	200	80	Pivot Right Turn
0x03				
0x02				
0x01				
0xC0	PIVOT_LEFT	80	200	Pivot Left Turn
0x40				
0x80				
0xE0				
0x00	STOP	0	0	Stop

Append your code to Lab 7

- When you receive a command byte to enable the line tracing mode,
  - `line_tracing = true;`
- When you receive a command byte to disable the line tracing mode,
  - `line_tracing = false;`
- In `loop()`, check the `line_tracing` flag
  - `if (line_tracing == true)`
    - Keep controlling the movement of SmartCAR

Part I: Start the line tracer

```
boolean line_tracing = false;

...

void line_tracing_enable()
{
    line_tracing = true;
    Serial.write("Line tracing is enabled..");
}

void line_tracing_disable()
{
    line_tracing = false;
    move_stop();
    Serial.write("Line tracing is disabled..");
}
```

```
void serialEvent()
{
    int command = Serial.read();

    switch (command)
    {
        ...
        case 12:
            line_tracing_enable();
            break;

        case 13:
            line_tracing_disable();
            break;
        default:
    }
}
```

- If the SmartCAR receives a byte of 12, enable line tracer
- If the SmartCAR receives a byte of 13, disable line tracer

Part I: Start the line tracer

```
void loop()
{
    if (line_tracing == true)
    {
        unsigned char sensor_data = 0;
        int z;

        for(z=0;z<8;z++)
        {
            unsigned int val = digitalRead(SensorD[z]);
            sensor_data |= (val << z);
        }
        sensor_data = ~sensor_data;

        Serial.print(sensor_data, HEX);
        Serial.write(" ");

        switch (sensor_data)
        {
            case 0x18:
            case 0x10:
            case 0x08:
            case 0x38:
            case 0x1c:
            case 0x3c:
              move_forward_speed(140, 140);
              break;
```

```
            case 0x0c:
            case 0x04:
            case 0x06:
            case 0x0e:
            case 0x1e:
            case 0x0f:
              turn_right_speed(200, 0);
              break;

            case 0x30:
            case 0x20:
            case 0x60:
            case 0x70:
            case 0x78:
            case 0xf0:
              turn_left_speed(0, 200);
              break;

            case 0x07:
            case 0x03:
            case 0x02:
            case 0x01:
              turn_pivot_right_speed(200, 80);
              break;
```

Part I: Start the line tracer

```
            case 0xc0:
            case 0x40:
            case 0x80:
            case 0xe0:
                turn_pivot_left_speed(80,  200);
                break;

            case 0x00:
            case 0xff:
                move_stop();
                break;

            default:
                move_stop();
                break;
        }
        delay(5);
    }
}
```

- Between each control,
  - Give a delay of 5ms
  - `delay(5);`
- Fill out the following functions

```
void move_forward_speed(int left, int right)
{
}
void turn_left_speed(int left, int right)
{
}
void turn_right_speed(int left, int right)
{
}
void turn_pivot_left_speed(int left, int right)
{
}
void turn_pivot_right_speed(int left, int right)
{
}
```

Today

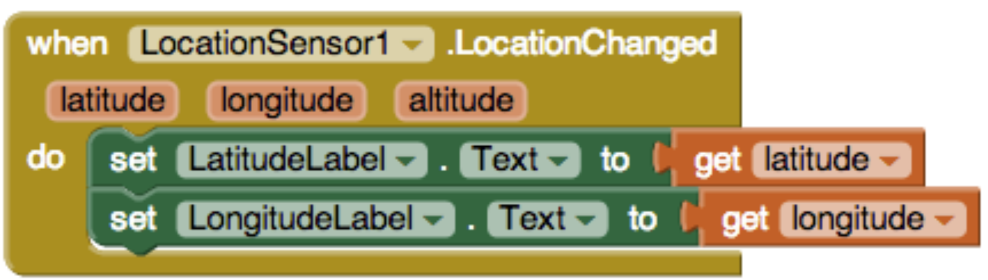
- Review from the last lecture
  - Infrared sensors

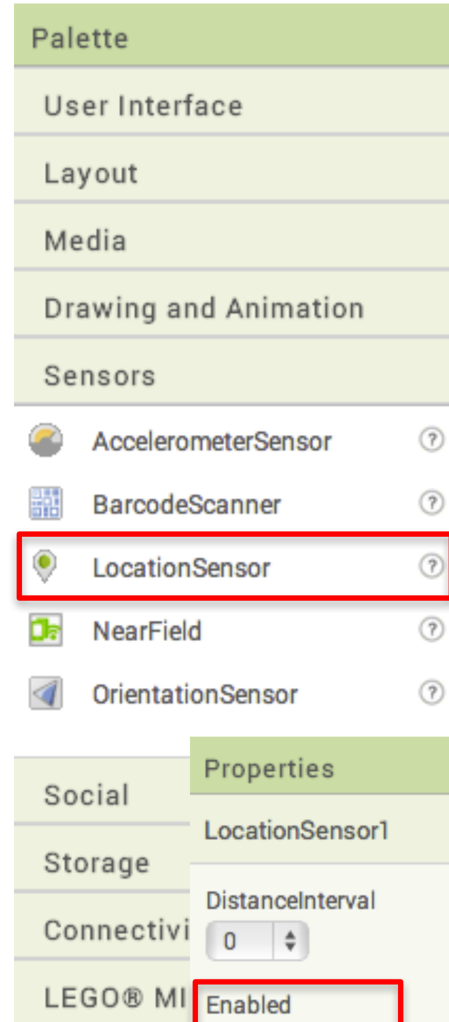
- Line tracing
- **Accessing location information from Android**

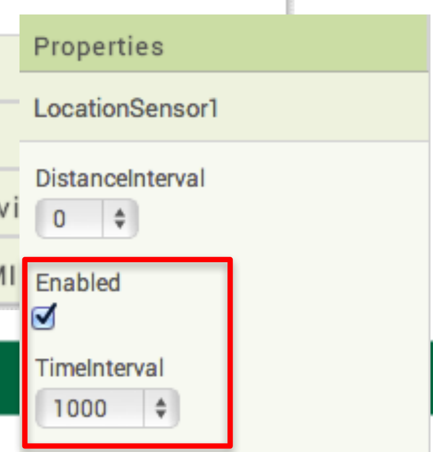
- Announcement

LocationSensor in App Inventor

- Under "Sensors"
  - Use "LocationSensor"
- Properties
  - Initially Enabled
  - Checking interval
    : "TimeInterval" set to 1000 (ms)
- When location has been changed,
  - "latitude", "longitude", "altitude" can be given to you







Course Announcement

- For lab session, we will cover
  - Will implement a line tracer

- Next week
  - Gyro, Accelerometer, and Compass sensors