**Embedded 중간고사**

* ASC 시리얼 연결 (Full duplex 방식, 송수신 동시 가능)
* 비동기식 전송 방식
* TX – 송신전용 신호선, 흰색, 8번
* RX – 수신전용 신호선, 초록, 10번
* start: 1bit – data: 8bit – parity: 0bit – stop: 1bit 포맷을 가짐
* I2C (Inter-Integrated Circuit) 버스 방식 (Half duplex 방식, 송수신 동시 불가)
* SDA: Serial DAta line – 데이터 비트의 신호선
* SCL: Serial Clock Line – 동기용 클럭 신호선
* SPI (Serial Peripheral Interface) 버스 방식 (Full duplex 방식, 송수신 동시 가능)
* MOSI: Master Output Slave Input – master에서 나와서 slave로 들어가는 데이터 신호선
* MISO: Master Input Slave Output - slave에서 나와서 master로 들어가는 데이터 신호선
* SCLK: Serial CLocK – 동기용 클럭 신호선
* SS(=CE): Slave Slect (=Chip Enable) – slave 칩을 선택해 주는 선
* 회로연결의 기본
* 0V에 – 꽂기, – 에는 GND 꽂기
* 5V, 3.3V에 + 꽂기
* Node.js 기반 임베디드 제어 코드 양식
* gpio.HIGH == 1, gpio.LOW == 0
* gpio.setup(‘wpi’) == gpio.wiringPiSetup();

const gpio = require(‘node-wiring-pi’);

const 핀넘버 = wpi넘버;

각종 변수 선언 및 함수 정의;

gpio.setup(‘wpi’);

gpio.pinMode(핀넘버, gpio.INPUT 혹은 gpio.OUTPUT);

함수 실행;

* 2색 LED 모듈

|  |  |  |
| --- | --- | --- |
| GND | DO | S |

* 파워 필요 없음.
* 0V를 –에 연결 후, GND와 연결
* DO와 GPIO 연결
* 3색 LED 모듈

|  |  |  |  |
| --- | --- | --- | --- |
| Blue | Green | Red | GND |

* 연결 방법은 2색과 상이
* LED 예제 코드

const TimeOutHandler = function() {

if (count > 0) {

gpio.digitalWrite(LEDPIN, 1);

console.log(“LED ON”);

count = 0;

}

else {

gpio.digitalWrite(LEDPIN, 0);

console.log(“LED OFF”);

count = 1;

}

setTimeout(TimeOutHandler, 1000); // 콜백함수

}

gpio.setup(‘wpi’);

gpio.pinMode(LEDPIN, gpio.OUTPUT);

setTimeout(TimeOutHandler, 1000);

* 버튼 모듈

|  |  |  |
| --- | --- | --- |
| GND | 5V | S |

* 0V를 –에 연결 후, GND와 연결
* 5V를 +에 연결 후, 5V와 연결
* S와 GPIO 연결
* 버튼 예제 코드

const CheckButton = function() {

let data = gpio.digitalRead(BUTTON);

if (!data) // 버튼이 눌렸을 때 data = 0 이므로, !data 사용

console.log(“BUTTON PRESSED”);

setTimeout(CheckButton, 300); // 0.3초에 한번씩 버튼 체크 확인

}

process.on(‘SIGINT’, function() {

console.log(“EXIT PROGRAM”);

process.exit();

});

gpio.setup(‘wpi’);

gpio.pinMode(BUTTON, gpio.INPUT);

setImmediate(CheckButton); // 즉시실행

* 버튼 예제 코드 2 - 3초간 눌렀을 때 동작시키기

var flag = false; // 아래는 버튼에 대해 누르거나 뗄 때 인터럽트 발생

var clock = null; // INT\_EDGE\_FALLING, INT\_EDGE\_RISING 등이 있다.

wpi.wiringPiISR(BUTTON, wpi.INT\_EDGE\_BOTH, function() {

if (!data) {

if (flag == false) {

flag = true; // 밑의 함수를 한 번만 실행시키고 if문 탈출

clock = setTimeout(eventHandler, 3000);

}

}

else {

flag = false;

clearTimeout(clock); // clock 호출한걸 취소시켜줌

}

});

function eventHander () {

let data = gpio.digitalRead(BUTTON);

if (!data) {

console.log(‘BUTTON PRESSED’);

}

}

* 부저 모듈

|  |  |  |
| --- | --- | --- |
| GND | 5V | S |

* 0V를 –에 연결 후, GND와 연결
* 5V 핀을 +에 연결 후, 5V와 연결
* S와 GPIO 연결
* 5V를 연결하지 않아도 소리가 남
* 부저 예제 코드

const TurnOn = function() {

gpio.digitalWrite(BUZZER, 1);

console.log(“BUZZER ON”);

setTimeout(TurnOff, 200);

}

const TurnOff = function() {

gpio.digitalWrite(BUZZER, 0);

console.log(“BUZZER OFF”);

setTimeout(TurnOn, 200);

}

gpio.setup(‘wpi’);

gpio.pinMode(BUZZER, gpio.OUTPUT);

setTimeout(TurnOn, 200);

* 부저 예제 코드 2 – 2~4옥타브 연주

var tones = [65, 73, 82, 87 , … , 523] // 2, 3, 4옥타브 주파수

var index = 0;

const TurnOn = function(){

gpio.softToneCreate(BUZZER);

if (index >= tones.length – 1) i = 0;

gpio.softToneWrite(BUZZER, tones[index++]);

console.log(“%d번째(frequency: %d)”, i, tones[index]);

setTimeout(TurnOn, 1000);

}

* 광센서 모듈

|  |  |  |
| --- | --- | --- |
| GND | 5V | S |

* 0V를 –에 연결 후, GND와 연결
* 5V를 +에 연결 후, 5V와 연결
* S와 GPIO 연결
* 아날로그 광센서 모듈

|  |  |  |  |
| --- | --- | --- | --- |
| V6 | GND | DO | AO |

* 연결 방법은 위와 상이
* 광센서 예제 코드

const CheckLight = function() {

let data = gpio.digitalRead(LIGHT);

if(!data) { // 버튼과 마찬가지의 맥락

console.log(“BRIGHT”);

}

else{

console.log(“DARK”);

}

setTimeout(CheckLight, 500);

}

gpio.setup(‘wpi’);

gpio.pinMode(LIGHT, gpio.INPUT);

setTimeout(CheckLight, 200);

* 터치 모듈

|  |  |  |  |
| --- | --- | --- | --- |
| AO | GND | 5V | DO |

* 0V를 –에 연결 후, GND와 연결
* 5V를 +에 연결 후, 5V와 연결
* gpio와 DO 연결
* 터치모듈 예제 코드

const CheckTouch = function() {

let data = gpio.digitalRead(TOUCH);

if(data) // 버튼, 광센서와 다르게 !data가 아닌 data를 사용

console.log(“TOUCHED”);

setTimeout(CheckTouch, 300);

}

process.on(‘SIGINT’, function() {

console.log(“EXIT PROGRAM”)

process.exit();

});

gpio.setup(‘wpi’);

gpio.pinMode(TOUCH, gpio.INPUT);

setTimeout(CheckTouch, 10);

* Relay 스위치 모듈

|  |  |  |
| --- | --- | --- |
|  | 3.3V | OUT |
| S | 5V | GND |

* S와 GPIO 연결
* 5V를 +에 연결 후, 5V와 연결
* 0V를 –에 연결 후, GND와 연결
* 위의 3.3V에 5V도 연결 가능하지만, LED가 타버리기 때문에 3.3V 연결
* OUT과 LED의 색깔과 연결
* LED의 GND는 –와 연결
* Relay 스위치 예제 코드

const TurnOn = function(){

gpio.digitalWrite(RELAY, 1);

console.log(“RELAY ON”);

setTimeout(TurnOff, 3000); // 3초간 전원 공급

}

const TurnOff = function(){

gpio.digitalWrite(RELAY, 0);

console.log(“RELAY OFF”);

setTimeout(TurnOn, 3000); // 3초간 전원 차단

}

gpio.wiringPiSetup();

gpio.pinMode(RELAY, gpio.OUTPUT);

setTimeout(TurnOn, 200);

* 사운드 센서 모듈

|  |  |  |  |
| --- | --- | --- | --- |
| AO | GND | 5V | DO |

* 0V를 –에 연결 후, GND와 연결
* 5V를 +에 연결 후, 5V와 연결
* gpio와 DO 연결
* 가변저항기로 low값 지정
* 사운드 센서 코드 예제

const DetectSound = function() {

let data = gpio.digitalRead(SOUND);

if(data) // touch와 함께 data 사용

console.log(“LOUD”);

setTimeout(DetectSound, 10);

}

process.on(‘SIGINT’, function() {

console.log(“PROGRAM EXIT”);

process.exit();

}

gpio.wiringPiSetup();

gpio.pinMode(SOUND, gpio.INPUT);

setTimeout(DetectSound, 10);

* polling과 event-driven방식 차이
* polling: cpu가 무한 루프를 돌며 사용자, 혹은 프로그램의 결과를 기다림. 해당 작업을 기다리는 동안에는 다른 코드를 실행할 수 없음
* event-driven: 특정 입력이나 작업 결과가 입력되면 특정 이벤트가 발생, 그 순간 해당 이벤트와 관련된 코드를 실행시킴.
* 라즈베리파이 GPIO핀을 초기화하는 함수
* setup(‘wpi’): wpi, gpio, sys, phys 설정 가능
* wiringPiSetup(): GPIO핀을 wpi 핀번호로 설정
* wiringPiSetupPhys(): GPIO핀을 물리 핀번호로 설정
* ADC칩 구성 (MCP3208) - SPI통신 사용
* 아날로그 센서 8개까지 연결 가능
* 아날로그 정보를 12bit로 변환해줌
* 위에서부터 3.3V Analog, 5V Analog, digital 순
* 3.3V, 5V, MOSI, MISO, SCLK, GND, CE0을 아날로그 칩과 똑같은 핀번호에 연결
* 아날로그 모듈은 위에서부터 AO, GND, V순으로 연결
* SPI를 이용한 아날로그 소리센서 측정 코드 예제

const gpio = require(‘node-wiring-pi’);

const mcpadc = require(‘mcp-adc’);

const soundsensor = new mcpadc.Mcp3208();

const CS\_MCP3208 = 10 // wpi 10번인 CE에 연결

const SPI\_CHANNEL = 0 // ADC 0번째 채널 선택 (0~11중 맨 위 첫 번째)

const SPI\_SPEED = 1350000 // 1.3Mhz

const QuietSound = 1935;

const SoundDetect = function() {

soundsensor.readRawValue(SPI\_CHANNEL, function(value){ // 마이크 센서에 대해!

if(value > QuietSound)

console.log(“소리인식 = %d”, value);

else console.log(“%d”, value);

});

setTimeout(SoundDetect, 500);

}

process.on(‘SIGINT’, function() {

console.log(“PROGRAM EXIT”);

process.exit();

});

gpio.wiringPiSetup();

gpio.wiringPiSPISetup(SPI\_CHANNEL, SPI\_SPEED); // 마이크 모듈에 대해 SPI setup!

gpio.pinMode(CS\_MCP3208, gpio.OUTPUT); // adc칩에 대해 핀 모드 설정!

setTimeout(SoundDetect, 500);

* data와 !data 정리
* button은 !data
* 광센서는 !data
* 터치는 data
* 사운드는 data