- Internal Battery **Voltage** = 3.6V
- Output **Voltage** = 5.1V (Via V boost/buck circuit)
- Output **Current** = 2.1A x 2 Channel (But sum max = 3.6A)
- Output **Power** = 5.1x2.1 = 10.71 Watts
- **Energy Capacity** (at 5.1V) = 10200 mAh
  = 10200x5.1 = 52.02 Wh

Credit iFixit

S20 ultra Battery Capacity
5000 mAh x 3.86V = 19.30 Wh

Phone supports 45W charging.
In theory it can charge from 0% to 100% in

19.30/45 = 0.43 hrs (25.8 min)

BUT wait …

You also need the following:

Charger that
- Supports PD 3.0, PPS 2.0
- Supports 10V/4.5A
- E-marked USB-C Cable

- เอาต์พุต Type-C: 5 V/3A, 9 V/3A, 12 V/3A, 14.5 V/3A, 20 V/3A (PD60W Max)
- เอาต์พุต USB-A1: 5 V/2.4A
- เอาต์พุต USB-A2: 5 V/2.4A

Most likely get this power output
9x3 = 27 Watt

Type-C cable must also handle this power

Power Supply?

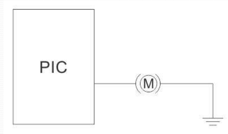
Max. current	0.5 A (USB 2.0) 0.9 A (USB 3.0 & 3.1) 3 A (USB Type-C)

Other types of batteries

Li-ion (Lithium ion)
Rechargeable, High capacity,
Explode when short circuited

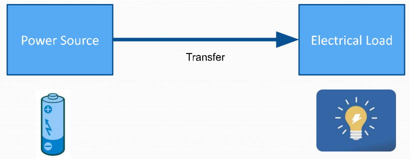
LIPO (Lithium Polymer)
Rechargeable, Very high current (Good for Drones)
Easily damaged

What's wrong with this circuit?



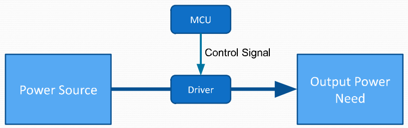
A microcontroller usually supplies limited current (i.e. 25 mA).
This is ok for LEDs but not enough for high current loads like a motor

Output Control: Power transfer



Driving an electrical load is essentially about transferring power from the source to the load.

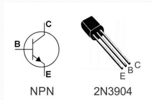
Output Control: Power transfer



The transfer is done using a "driver" which is a device that, for this course, can be controlled by a micro-controller.

Transistors

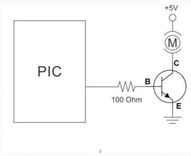
Using transistors as a switch



Transistors works both as (1) a switch and (2) a current amplifier.
Transistors allow us to send higher current to a load.

Using a transistor

Low-side switching



100 Ohm resistor is to limit the amount of current triggering the base

Different Loads Requires Different Power

Typical LEDs

5-15 mA @5V
Or 25-75 mW

Small DC Motors

100-200 mA @5V
Or 0.5-1W

Irrigation Control Valve

0.6-1.0 A @24V
Or 14-24W

Calculate the Power Parameters

0.6-1.0 A @24V

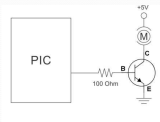
Voltage = 24V

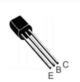
Current = 0.6-1.0A

Power = 24x1 = 24W

Energy = Power * Time

Can the 2N3904 control the valve?



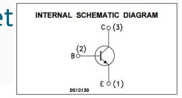


2N3904

Irrigation Control Valve

0.6-1.0 A @24V
Or 14-24W

The answer is in the datasheet



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{CBO}	Collector-Base Voltage ($I_E = 0$)	60	V
V_{CEO}	Collector-Emitter Voltage ($I_B = 0$)	40	V
V_{EBO}	Emitter-Base Voltage ($I_C = 0$)	6	V
I_C	Collector Current	200	mA
P_{tot}	Total Dissipation at $T_C = 25\ ^oC$	625	mW
T_{stg}	Storage Temperature	-65 to 150	^oC
T_j	Max. Operating Junction Temperature	150	^oC

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{CBO}	Collector-Base Voltage ($I_E = 0$)	60	V
V_{CEO}	Collector-Emitter Voltage ($I_B = 0$)	40	V
V_{EBO}	Emitter-Base Voltage ($I_C = 0$)	6	V
I_C	Collector Current	200	mA
P_{tot}	Total Dissipation at $T_C = 25\ ^oC$	625	mW
T_{stg}	Storage Temperature	-65 to 150	^oC
T_j	Max. Operating Junction Temperature	150	^oC

1. Control 24V? -> Yes
2. Can supply enough current? -> No.

- At 24V, the maximum current is 625/24 = 26 mA
- Notice that 26 mA is a lot less than 200 mA
- 200 mA can be achieved only at 625/200 = 3.125 V

How about this one?

KTD882

MAXIMUM RATINGS (Ta=25℃)

CHARACTERISTIC		SYMBOL	RATING	UNIT
Collector-Base Voltage		V_{CBO}	40	V
Collector-Emitter Voltage		V_{CEO}	30	V
Emitter-Base Voltage		V_{EBO}	5	V
Collector Current	DC	I_C	3	A
	Pulse (Note)	I_{CP}	7	
Base Current (DC)		I_B	0.6	A
Collector Power Dissipation	Ta=25℃	P_C	1.5	W
	Tc=25℃		10	
Junction Temperature		T_j	150	℃
Storage Temperature Range		T_{stg}	−55~150	℃

Note : Pulse Width ≤10mS, Duty Cycle≤50%.

MAXIMUM RATINGS (Ta=25℃)

CHARACTERISTIC		SYMBOL	RATING	UNIT
Collector-Base Voltage		V_{CBO}	40	V
Collector-Emitter Voltage		V_{CEO}	30	V
Emitter-Base Voltage		V_{EBO}	5	V
Collector Current	DC	I_C	3	A
	Pulse (Note)	I_{CP}	7	
Base Current (DC)		I_B	0.6	A
Collector Power Dissipation	Ta=25℃	P_C	1.5	W
	Tc=25℃		10	
Junction Temperature		T_j	150	℃
Storage Temperature Range		T_{stg}	−55~150	℃

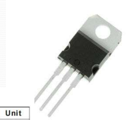
Note : Pulse Width ≤10mS, Duty Cycle≤50%.

1. Control 24V? -> Yes
2. Can supply enough current? -> No.

- At 24V, the maximum current is 10/24 = 0.416 A

Ta = Atmosphere Temperature
Tc = Case Temperature

This one? ST-13005



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{CES}	Collector-Emitter Voltage (V_{BE} = 0)	700	V
V_{CEO}	Collector-Emitter Voltage (I_B = 0)	400	V
V_{EBO}	Emitter-Base Voltage (I_C = 0)	9	V
I_C	Collector Current	4	A
I_{CM}	Collector Peak Current (t_p < 5 ms)	8	A
I_B	Base Current	2	A
I_{BM}	Base Peak Current (t_p < 5 ms)	4	A
P_{tot}	Total Dissipation at T_c = 25 °C	75	W
T_{stg}	Storage Temperature	-65 to 150	°C
T_j	Max. Operating Junction Temperature	150	°C

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{CES}	Collector-Emitter Voltage (V_{BE} = 0)	700	V
V_{CEO}	Collector-Emitter Voltage (I_B = 0)	400	V
V_{EBO}	Emitter-Base Voltage (I_C = 0)	9	V
I_C	Collector Current	4	A
I_{CM}	Collector Peak Current (t_p < 5 ms)	8	A
I_B	Base Current	2	A
I_{BM}	Base Peak Current (t_p < 5 ms)	4	A
P_{tot}	Total Dissipation at T_c = 25 °C	75	W
T_{stg}	Storage Temperature	-65 to 150	°C
T_j	Max. Operating Junction Temperature	150	°C

1. Control 24V? -> Yes
2. Can supply enough current? -> Yes.

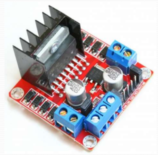
- At 24V, the maximum current is 75/24 = 3.125 A
- 75W can be achieved only when Tc (case) is 25C.

Higher loads

Drone Motor
14-25A

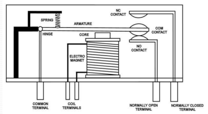
Motor Driver Modules

Complex and more expensive but easy to use.



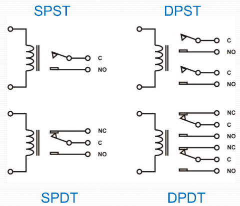
Current capacity 20-40 Amps typical

Relays



Usually have higher current ratings for its price.
15-50 Amps @5V typical for less than 100 THB

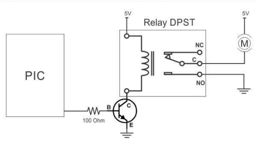
Relay "Poles" and "Throws"

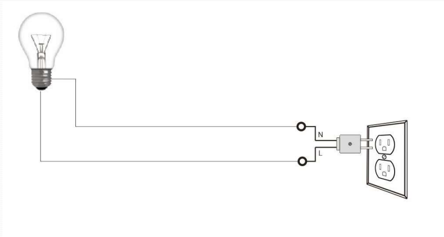


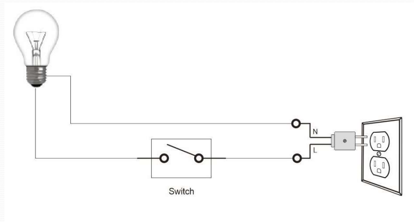
Poles = How many circuits are there?
Throws = Either NC or NO or both NC/NO

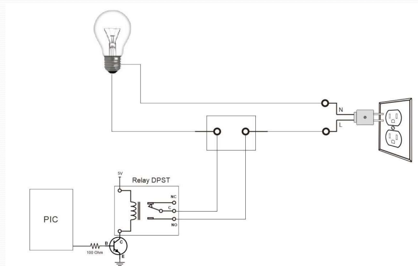
Note:
NC = Normally Closed
NO = Normally Opened

Driving load with a relay

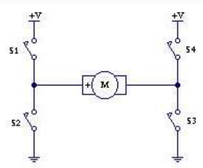


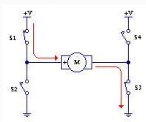


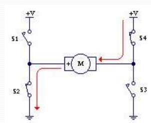




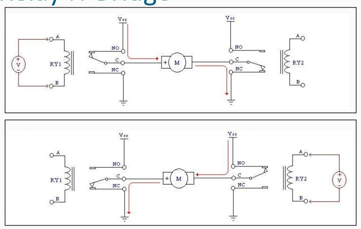
H-Bridge: A bi-directional driver



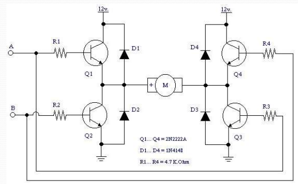


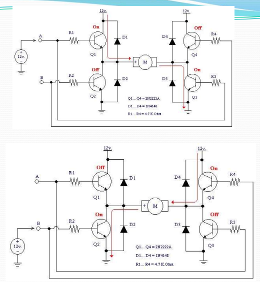


Relay H-Bridge



Transistor H-Bridge

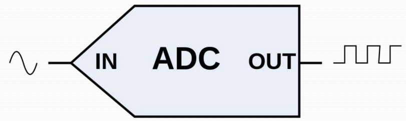




Conclusions

- There are many ways to drive output devices
- Transistors are good for low power applications
- Relays are suitable for AC power applications or when low loss and high current is needed
- H-bridge is for bi-directional applications
- Choose the right tool for the job!

1. หลักการของ ADC

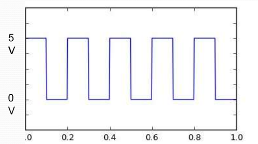


สัญญาณ
Analog (Voltage)

ค่า Digital

1.1 Sampling Rate Case Study: Speed Clicker Game

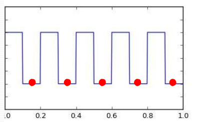
Player tries to press a button as many times as possible within a given time.1.1



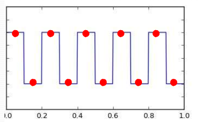
What is the minimum sampling rate in order to catch all the clicks?
Pulse period = 0.2 sec -> Freq = 1/0.2 = 5 Hz

Nyquist Frequency

- Minimum sampling rate must be **2x the signal rate**.
- Slower sampling rates can cause "aliasing"

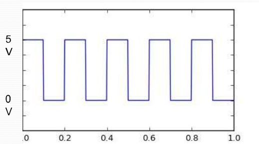


5Hz sampled output misses all the pulses



10Hz sampled output will capture the pulses

1.2 ADC Resolution



What is the minimum ADC resolution needed for this application
A Square wave has only two values: high and low
Thus, a 1 bit ADC is sufficient

How about the **ESP32** mcu?

4.1.2 Analog-to-Digital Converter (ADC)

ESP32 integrates 12-bit SAR ADCs and supports measurements on 18 channels (analog-enabled pins).

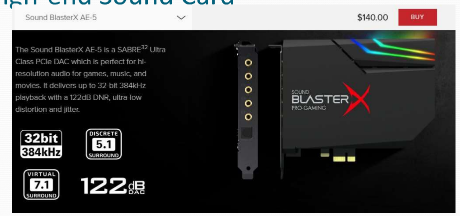
Table 7 describes the ADC characteristics.

Table 7: ADC Characteristics

Parameter	Description	Min	Max	Unit
DNL (Differential nonlinearity)	RTC controller; ADC connected to an external 100 nF capacitor; DC signal input; ambient temperature at 25 °C; Wi-Fi&BT off	–7	7	LSB
INL (Integral nonlinearity)		–12	12	LSB
Sampling rate	RTC controller	-	200	ksps
	DIG controller	-	2	Msps

Sampling rate is very good.
But the ADC resolution is not very high for audio.

High-end Sound Card

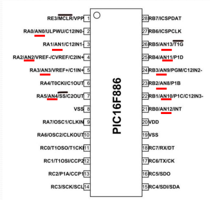


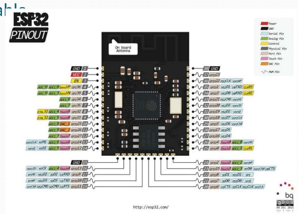
3. Number of ADC ports

Device	Program Memory	Data Memory		I/O	10-bit A/D (ch)	ECCP/ CCP	EUSART	MSSP	Comparators	Timers 8/16-bit
	Flash (words)	SRAM (bytes)	EEPROM (bytes)							
PIC16F882	2048	128	128	24	11	1/1	1	1	2	2/1
PIC16F883	4096	256	256	24	11	1/1	1	1	2	2/1
PIC16F884	4096	256	256	35	14	1/1	1	1	2	2/1
PIC16F886	8192	368	256	24	11	1/1	1	1	2	2/1
PIC16F887	8192	368	256	35	14	1/1	1	1	2	2/1

Which pins are ADC pins?

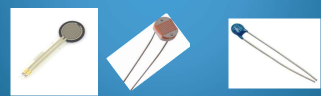
Not all pins are analog-capable





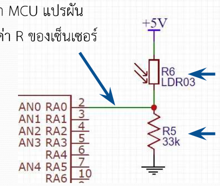
2. วงจรแบ่งแรงดัน (Voltage Divider)

บ่อยครั้งที่ sensor เป็นตัวต้านทาน แต่ ADC วัดได้แค่แรงดันไฟฟ้า
เราจะเปลี่ยนความต้านทานให้เป็นแรงดันได้อย่างไร

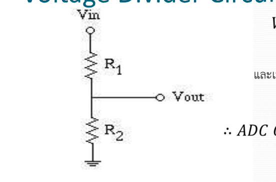


ตัวอย่างวงจรแบ่งแรงดัน

V เข้า MCU แปรผัน
ตามค่า R ของเซ็นเซอร์



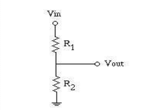
Voltage Divider Circuit



$$Vout = \frac{R2}{R1+R2} . Vin$$

และเนื่องจาก $ADC\ Output\ \alpha\ Vout$

$$\therefore ADC\ Output = \frac{R2}{R1+R2} . Max\ ADC\ Value$$



ถ้า R2 = 33K, Vin = 5V
ถ้า Rsensor (R1) = 99K

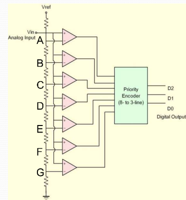
- ค่าที่อ่านจาก ADC จะเป็นเท่าใด? (1.25V)
- ค่า ADC Output จะเป็นเท่าใด? (255 ถ้าใช้ ADC 10 bit)
- ถ้าสลับตำแหน่ง Rsensor กับ R อ้างอิงคำตอบข้างต้นจะเปลี่ยนแปลงไปอย่างไร

3. ADC Conversion Methods

(เนื้อหาเสริม)

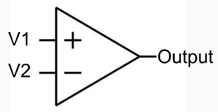
- Direct conversion (Flash ADC)
- Ramp-compare ADC
- Successive-Approximation

1. Direct Conversion



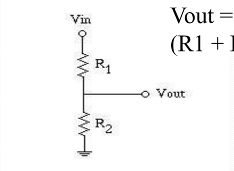
แรงดันอ้างอิง (Vref) จะถูกแบ่งค่าลงเรื่อยๆ และป้อนเข้าชุดวงจรเปรียบเทียบแรงดัน
วงจรเปรียบเทียบใดที่ Vin มากกว่า Vref ก็จะให้ค่าเป็น 1 วงจรที่เหลือจะให้ค่าเป็น 0

OpAmp as a Comparator



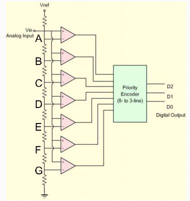
Input	Output
$V_1 > V_2$	High (1)
$V_1 < V_2$	Low (0)

Voltage Divider



Vout = Vin * R2 / (R1 + R2)

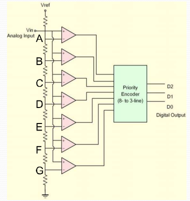
Direct Conversion



V at A = Vref * 7R / 8R
V at B = Vref * 6R / 8R
...
V at G = Vref * R / 8R

V at A = Vref * 7/8
V at B = Vref * 6/8
...
V at G = Vref * 1/8

Direct Conversion

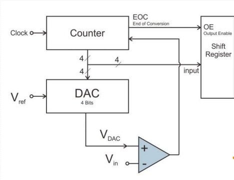


If Vref = 8V and Vin = 5.5 V
What is the output of the ADC?

Output of comparator = 0b0011111
Output of encoder = 0b101

Demo

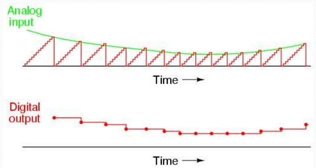
2. Ramp-Compare ADC



Demo

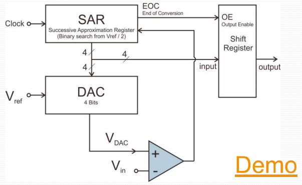
ใช้วงจรเปรียบเทียบแรงดันเพียงชุดเดียว โดยมีวงจรสร้างดัน (DAC) ค่อยๆ เพิ่มแรงดัน
เปรียบเทียบขึ้นจนมีค่าเกิน Vin

Ramp-Compare ADC Signal Example



ข้อเสียอย่างหนึ่งของ Ramp-Compare คือใช้เวลาในการแปลงค่าไม่เท่ากัน Analog
Input ที่มีค่ามากจะใช้เวลาแปลงนานกว่า ซึ่งเวลาที่แตกต่างกันนี้อาจส่งผลต่อจังหวะ
การทำงานของ MCU

3. Successive-Approximation



Demo

คือ Ramp Compare ADC ที่ปรับปรุงโดยใช้หลักของ Binary Search ในการหาค่าที่
ใกล้เคียง Vin ที่สุด (แทนการค่อยๆ เพิ่มค่าจากน้อยไปมาก)

ปัญหาของการใช้ threshold ค่าเดียวในการควบคุม

```
while(true) {
    If sensor1 < 500 [ turnOn(); ]
    If sensor1 > 500 [ turnOff(); ]
}
```

ปัญหาของการใช้ threshold ค่าเดียวในการควบคุม (ต่อ)

- ค่าเซ็นเซอร์จำนวนมากมีการแกว่ง
- บางครั้งเกิด Feedback จาก output เช่น
  - ปั๊มน้ำอัตโนมัติ: ลูกลอยไม่อยู่นิ่งเพราะน้ำกระเพื่อมจากน้ำที่ปั๊มสูบเข้า
  - โคมไฟอัตโนมัติ: แสงจากหลอดไฟบางส่วนวกกลับไปตกกระทบเซ็นเซอร์แสง

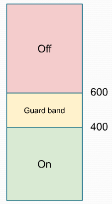
การติดดับติดต่อกันมักส่งผลเสียต่ออุปกรณ์

- เกิดไฟกระชากบ่อยครั้งหากเป็นอุปกรณ์ที่กินพลังงานมาก เช่นปั๊มน้ำ รีเลย์
- อายุการใช้งานของอุปกรณ์สั้นลง
- ส่งผลให้ผู้ใช้เกิดความรำคาญ กระทบต่อการใช้งานจริง [ตัวอย่าง 7-Eleven]

การใช้ Hysteresis แก้ไขปัญหา

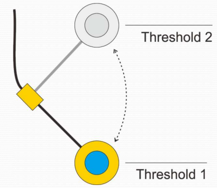
กำหนด on threshold และ off threshold แยกกัน

```
while(ture) {
  If sensor1 < 400 [ turnOn(); ]
  If sensor1 > 600 [ turnOff(); ]
}
```



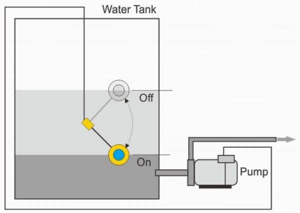
บางครั้งเซ็นเซอร์มี hysteresis ในตัว

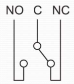
ลูกลอยไฟฟ้า



สวิทต้านในจะสับและปล่อยที่ระดับต่างกัน

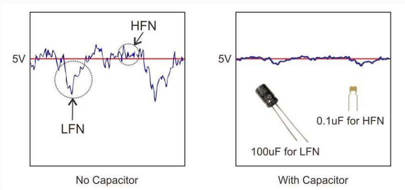
Typical Use: Water Pump Control



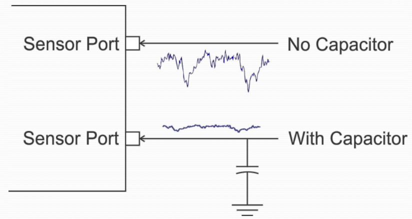


Sensor Pinout

Simple Low-Pass Filtering: Hardware Filtering using Capacitors

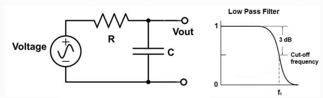


Capacitors slow down the signal change



How do we know the right capacitor size to use?

1. Trial & error
2. Calculate the frequency response of a low-pass filter



$$f_c = \frac{1}{2\pi RC}$$

Fc is where the output power (not amplitude) is half of the input.

Digital Interfaces and Basic PIC-C I2C Communication

Arnan (Roger) Sipitakiat
Department of Computer Engineering, Chiang Mai University, Thailand

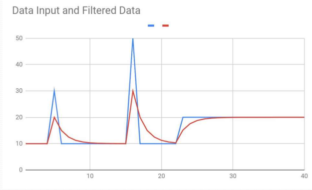
Simple Low-Pass Filtering: Software Implementation

$$X_n = \frac{wX_{(n-1)} + I}{w+1}$$

Where
- Xn = The current average value
- X(n-1) = The previous average value
- I = new sensor value
- w = weight

Example: Small Weight

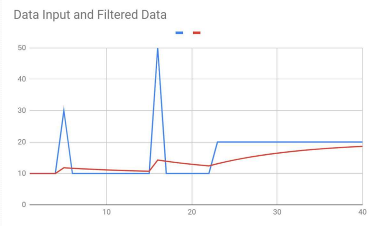
Limited noise rejection



Open this example

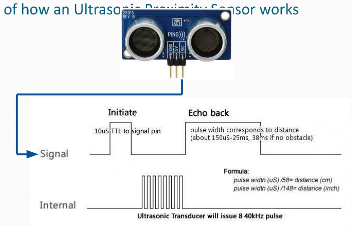
Example: Higher weight

Better noise rejection, but also causes a delay



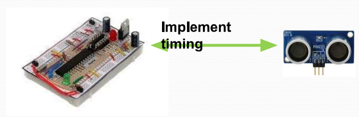
Why I2C?

Case study of how an Ultrasonic Proximity Sensor works



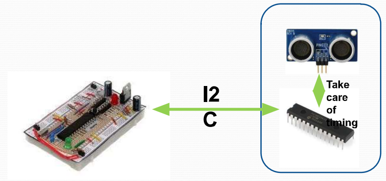
Direct Connection to the module

You need to do all the timing yourself



Offload to an I2C Host

Let an I2C host perform the timing

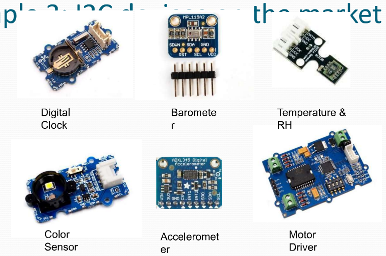


This idea is similar to software abstraction where you create subroutines to hide unnecessary details

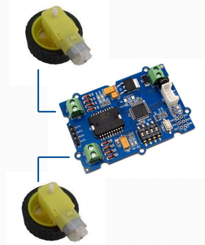
Tradeoffs of offloading

- **Benefits**
  - The sensor is simpler to use.
  - The main processor is free to do other things.
  - More modular.
- **Drawbacks**
  - More expensive
  - Must be careful of timing problems on the I2C host

Example 2: I2C devices on the market



Motor Driver Module



Register Address	Function
0x82	Speed (0-255)
0x84	PWM Frequency
0xAA	Direction (0/1)
0xA1	Select Motor A (0/1)
0xA5	Select Motor B (0/1)

Clock Module

Module address = 0xD0

Table 2. Timekeeper Registers

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12 / 24	10 Hour / PM/AM	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

0 = Always reads back as 0.

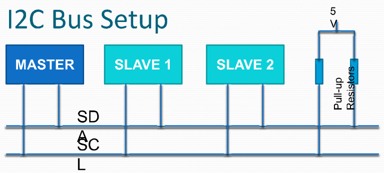
The DS1307 can be run in either 12-hour or 24-hour mode. Bit 6 of the hours register is defined as the 12-hour or 24-hour mode-select bit. When high, the 12-hour mode is selected. In the 12-hour mode, bit 5 is the AM/PM bit with logic high being PM. In the 24-hour mode, bit 5 is the second 10-hour bit (20 to 23 hours). The hours value must be re-entered whenever the 12/24-hour mode bit is changed.

Bitwise Operations

I2C Basics

- I2C is a Bus. Serial is point-to-point
- Master and Slave pair
  - Master must start the communication
  - Slave can only respond
- Slave has a 8-bit address
  - Bit 0 is reserved for direction control

Basic I2C Bus Setup



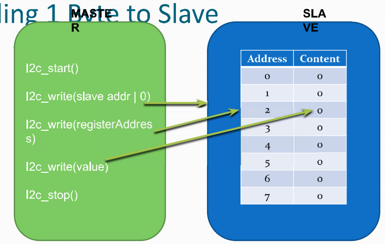
i2C Commands in PIC-C

Master Only
- I2c_start()
- I2c_stop()

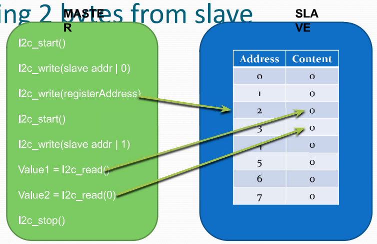
Slave Only
- I2c_isr_state()

Master & Slave
- I2c_read()
- I2c_write()

Sending 1 Byte to Slave



Reading 2 bytes from slave



Write ค่า 50 ไปยัง Register 5 ของ slave ที่มี address = 100

I2c_start();

Write ค่า 1000 ไปยัง Register 5 (ไบท์บน), 6 (ไบท์ล่าง) ของ slave ที่มี address = 100

I2c_start();

Read ค่าจาก Register 5 ของ slave ที่มี address = 100

Int returnValue;
I2c_start();

Read ค่าเซ็นเซอร์ 2 byte จาก Register 5,6 ของ slave ที่มี address = 100 และนำมาประกอบกันเป็นค่า int 16 Bit Register 5 = byte บน

Int16 Value;
I2c_start();

Example I
Controlling the Display Module
Address = 0xB0

Register Address	Function
2	High Byte
3	Low Byte

Example Program

```
// I2C - Master
#use i2c(MASTER, I2C1)

// Show the number 100 on the screen
I2c_start();
I2c_write(0xB0 | 0);   // display module address
I2c_write(2);   // show number command
I2c_write(0);
I2c_write(100);
I2c_stop();
```

Example II
Reading from the ultrasonic sensor

What is bitwise operations?

Logical Operation

3 and 2 = ?
3 && 2 = ?

Bitwise Operation

3 & 2 = ?

Bitwise vs Logical Operators

- Logical operators results in "True" or "False" only.
- Examples
  - 0b011 & 0b110 = 0b010 (bitwise)
  - 0b011 && 0b110 = True (logical)
  - ~0b010 = 0b101 (bitwise)
  - !0b010 = False (logical)

เลขฐาน 10, 2, และ 16

10 (ฐาน 10) = 0b1010 (ฐาน 2) = 0x0A (ฐาน 16)

ฐาน 2 ใช้สัญลักษณ์ตัว b ส่วน ฐาน 16 ใช้ตัว x

ฐาน 16 หนึ่งหลัก = 4 หลักฐาน 2

0x12 = 0b0001 0010

Useful Bitwise Operations

- Shift left, right (<<, >>)   0b01 << 1 = 0b10
- Bitwise AND (&)   0b10 & 0b01 = 0b00
- Bitwise OR (|)   0b01 | 0b10 = 0b11
- XOR (^)   0b11 | 0b10 = 01
- Bitwise Not (~)   ~0b01 = 0b10 = 0b11111110 (if using 8-bit)

Practice in Python using repl.it

repl.it

Clock Module

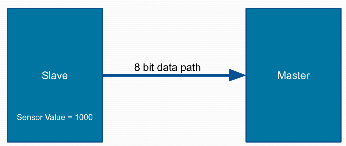
Table 2. Timekeeper Registers

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12 / 24	10 Hour / PM/AM	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

0 = Always reads back as 0.

The DS1307 can be run in either 12-hour or 24-hour mode. Bit 6 of the hours register is defined as the 12-hour or 24-hour mode-select bit. When high, the 12-hour mode is selected. In the 12-hour mode, bit 5 is the AM/PM bit with logic high being PM. In the 24-hour mode, bit 5 is the second 10-hour bit (20 to 23 hours). The hours value must be re-entered whenever the 12/24-hour mode bit is changed.

ถ้าค่าเซ็นเซอร์ = 1000 (10 bit) จะส่งค่านี้จาก slave ไป master ต้องเขียนโปรแกรมอย่างไรถ้าส่งข้อมูลได้ทีละ 1 byte (8 bit)



Write(1000);
???

ส่งค่า 16 bit ผ่าน bus ขนาด 8 bit โดยใช้ Bitwise Opertations

ฝั่งส่ง

~~write(1000); ผิด~~

Write(1000 & 0xFF); ไบท์ล่าง

Write (1000 >> 8); ไบท์บน

ฝั่งรับ

Int16 Value = read() + (read() << 8);

ได้เป็น value = 232 + (3 << 8) = 232+768 = 1000

พิจารณาค่าตัวแปรในรูปเลขฐานต่างๆ

Int16 **Data** = 1000;

1000 = 0x3E8 (base 16) = 0b1111101000 (base 2)

ไบท์บน | ไบท์ล่าง

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Bit ที่
0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	

Shifting Bits

Int16 **Data** = 0x3E8;

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Bit ที่
0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	

Data >> 8 = 3

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Data << 8 = 0xE8

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0

Bitwise AND

Int16 **Data** = 0x3E8;

Data & 0xFF00 = 0x300

0x03E8	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	&
0xFF00	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	
0x300	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	

Bitwise OR

Int16 **Data** = 0x3E8;

Data | 0xFF00 = 0xFFE8

0x03E8	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	|
0xFF00	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	
0xFFE8	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	

Bitwise NOT

Int16 **Data** = 0x3E8;

Data = ~Data

0x3E8	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	|
0xFC17	1	1	1	1	1	1	0	0	0	0	0	1	0	1	1	1	

What is bitwise operations?

Logical Operation

```
3 and 2 = ?
3 && 2 = ?
```

Bitwise Operation

```
3 & 2 = ?
```

Bitwise vs Logical Operators

- Logical operators results in "True" or "False" only.
- Examples
  - `0b011 & 0b110 = 0b010     (bitwise)`
  - `0b011 && 0b110 = True     (logical)`
  - `~0b010 = 0b101     (bitwise)`
  - `!0b010 = False     (logical)`

เลขฐาน 10, 2, และ 16

10 (ฐาน 10) = 0b1010 (ฐาน 2) = 0x0A (ฐาน 16)

ฐาน 2 ใช้สัญลักษณ์ตัว b ส่วน ฐาน 16 ใช้ตัว x

ฐาน 16 หนึ่งหลัก = 4 หลักฐาน 2

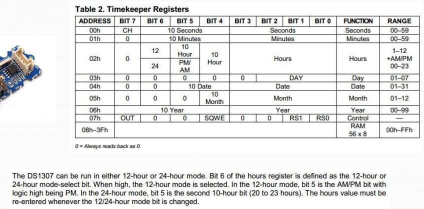
0x12 = 0b<u>0001</u> <u>0010</u>

Useful Bitwise Operations

```
• Shift left, right (<<, >>)   0b01 << 1 = 0b10
• Bitwise AND  ( & )     0b10 & 0b01 = 0b00
• Bitwise OR ( | )       0b01 | 0b10 = 0b11
• XOR (^)          0b11 | 0b10 = 01
• Bitwise Not ( ~ )         ~0b01 = 0b10
                  = 0b11111110
                  (if using 8-bit)
```

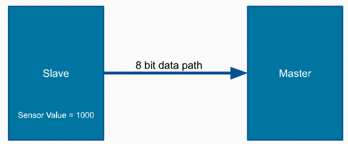
Practice in Python using repl.it

Clock Module



ถ้าค่าเซ็นเซอร์ = 1000 (10 bit) จะส่งค่านี้จาก slave ไป master

ต้องเขียนโปรแกรมอย่างไรถ้าส่งข้อมูลได้ทีละ 1 byte (8 bit)



Write(1000);
???

ส่งค่า 16 bit ผ่าน bus ขนาด 8 bit โดยใช้ Bitwise Opertations

ฝั่งส่ง

~~write(1000); ผิด~~

Write(1000 & 0xFF); ไบท์ล่าง

Write (1000 >> 8); ไบท์บน

ฝั่งรับ

Int16 Value = read() + (read() << 8);

ได้เป็น value = 232 + (3 << 8) = 232+768 = 1000

พิจารณาค่าตัวแปรในรูปเลขฐานต่างๆ

Int16 **Data** = 1000;

1000 = 0x3E8 (base 16) = 0b1111101000 (base 2)

ไบท์บน | ไบท์ล่าง

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Bit ที่
0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	

Shifting Bits

Int16 **Data** = 0x3E8;

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Bit ที่
0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	

Data >> 8 = 3

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Data << 8 = 0xE8

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0

Bitwise AND

Int16 **Data** = 0x3E8;

Data & 0xFF00 = 0x300

0x03E8	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	&
0xFF00	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	
0x300	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	

Bitwise OR

Int16 **Data** = 0x3E8;

Data | 0xFF00 = 0xFFE8

0x03E8	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0
0xFF00	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
0xFFE8	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0

Bitwise NOT

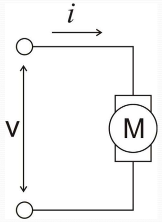
Int16 **Data** = 0x3E8;

Data = ~Data

0x3E8	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0
0xFC17	1	1	1	1	1	1	0	0	0	0	0	1	0	1	1	1

1. PWM คืออะไร?

กรณีศึกษา
สมมุติว่าต้องการควบคุมความเร็วของมอเตอร์
จะต้องออกแบบวงจรอย่างไร?

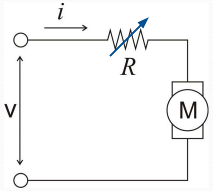


ตามหลักการทั่วไป

ความเร็วการหมุนขึ้นอยู่กับกำลังไฟฟ้าที่ป้อนให้กับมอเตอร์ ซึ่ง

Power = I x V

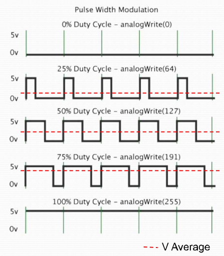
ซึ่งหากความต่างศักดิ์คงที่ สิ่งที่ต้องเปลี่ยนคือกระแส



วิธีหนึ่งที่ทำได้คือควบคุมกระแสด้วยตัวต้านทาน หรือตัวต้านทานปรับค่าได้ (Potentiometer หรือ POT)

แต่การเปลี่ยนค่าความต้านทานมักทำไม่สะดวกนัก โดยเฉพาะเมื่อต้องการควบคุมผ่านโปรแกรม

นอกจากนั้นโหลดที่ต้องใช้กระแสสูงก็มักทำให้เกิดความร้อนขึ้นที่ตัวต้านทานสูง



Pulse Width Modulation (PWM)

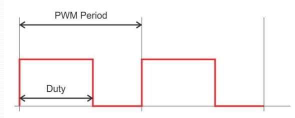
คือการสร้าง Pulse ที่ความถี่สูง* มีเวลาติดและดับแปรผันตามระดับพลังงานที่ต้องการจะส่งออกไปยังโหลด

เป็นทางออกที่ได้รับความนิยมเพราะ
- มักไม่ต้องแก้ไขวงจร
- ปรับพลังงานผ่านโปรแกรมได้

* คำเป็นเท่าไหร่เดี๋ยวดังาน

PWM Parameters

คุณสมบัติของ PWM ที่สำคัญคือ
- คาบ มีหน่วยเป็นเวลา
- Duty มีหน่วยเป็นเปอร์เซ็นต์



Period unit = time (e.g. ms)

Duty unit = % (0 to 100)

2. การใช้งาน PWM ในระบบสมองกลฝังตัว

2.1 Power - ใช้ควบคุมพลังงานที่ส่งไปยังโหลด

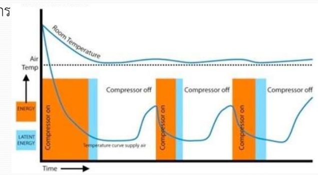
2.2 Signal – ใช้รับและส่งสัญญาณดิจิทัล

2.3 Digitize – แปลงค่าสัญญาณ อนาล็อก กับ ดิจิทัล

2.1 Power - ใช้ควบคุมพลังงานที่ส่งไปยังโหลด

เครื่องปรับอากาศ (แอร์)

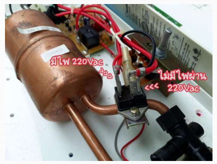
- Compressor ทำงานเมื่อต้องการลดอุณหภูมิ
- คาบและ Duty การทำงานของ Compressor เป็นเท่าใด?



กราฟสีเทาแสดงเวลาติด/ดับของ compressor

เครื่องทำน้ำอุ่น

- Triac ควบคุมการเร่งหรี่ไฟ
- ทำงานเท่ากับความถี่ของไฟบ้าน (50Hz)



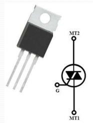
คาบการทำงานของ PWM ควรเป็นเท่าใด

ให้ดูที่ response time ของ Output นั้นๆ

- เตาแม่เหล็กไฟฟ้า
  Response ของอุณหภูมิเป็นหน่วยวินาที ดังนั้นคาบ PWM ~15-30 วินาที
- เครื่องปรับอากาศ
  Response อุณภูมิห้องหน่วยเป็นนาที ดังนั้นคาบ PWM ~1+ นาที
- หม้อทำน้ำอุ่น
  ใช้ Traic ควบคุมไฟ AC ที่ความถี่ 50Hz
- LED?
  Response ของแสงหน่วยเป็น ? ดังนั้นคาบ PWM ควรเป็น ?

การออกแบบวงจรควบคุมที่เหมาะกับคาบที่ต้องการ

Transistor (for DC)



Triac (for AC)

Relay (DC/AC)

อายุการใช้งาน Relay

สมมุติอายุการใช้งาน 100,000 operations

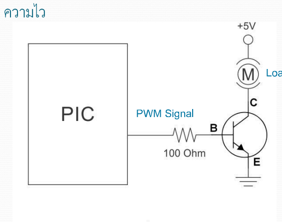
ถ้าใช้กับ PWM ที่มีคาบ = 10 Hz

อายุการใช้งานของ relay จะนานเท่าใด?

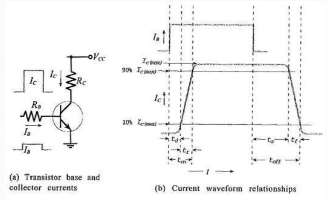
คำตอบ: 100,000/10 = 10,000 วินาที = 2.78 ชม

Transistor มีอายุการใช้งานนานมาก แต่มีข้อจำกัดอย่างอื่น

เช่น กระแส, ความไว



ตัวอย่างข้อจำกัดในการสร้าง PWM ของ Transistor



Td = Delay Time

Tr = Rising Time

Ts = Storage Time

Tf = Falling Time

Case Study: NPN 2N3904 Transistor

SWITCHING CHARACTERISTICS

Delay Time	(V_{CC} = 3.0 Vdc, V_{BE} = 0.5 Vdc, I_C = 10 mAdc, I_{B1} = 1.0 mAdc)		t_d	–	35	ns
Rise Time			t_r	–	35	ns
Storage Time	(V_{CC} = 3.0 Vdc, I_C = 10 mAdc, I_{B1} = I_{B2} = 1.0 mAdc)	2N3903 2N3904	t_s	– –	175 200	ns
Fall Time			t_f	–	50	ns

Pulse ที่สั้นที่สุด = Rise Time + Storage Time + Fall Time

= 35+200+50 = 285 ns

ถ้าสมมุติว่า PWM ที่ต้องการมี 100 ระดับ แสดงว่า 285 ns คือ 1% ของคาบ

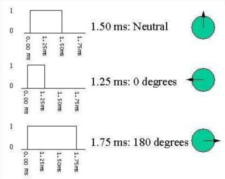
ดังนั้นคาบเท่ากับ 285x100 = 28,500 ns หรือ 35.088 KHz

2.2 Signal – ใช้ PWM รับและส่งสัญญาณดิจิทัล

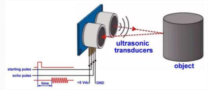
Servo Motor ใช้ PWM เป็น Input
กำหนดตำแหน่งของมอเตอร์

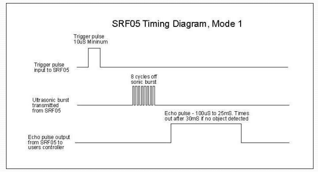
Ultrasonic Proximity Sensor
ส่งค่าระยะที่วัดได้ออกมาเป็น PWM

Servo Motor Control

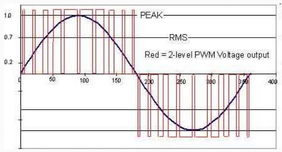


Ultrasonic Proximity Signal



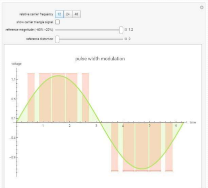


2.3 Digitize – แปลงค่าสัญญาณ อนาล็อก กับ ดิจิทัล



ตัวอย่างการแปลงสัญญาณอนาล็อกเป็น
ดิจิทัลโดยใช้ PWM

Live Demo



https://demonstrations.wolfram.com/PulseWidthModulationPrinciple