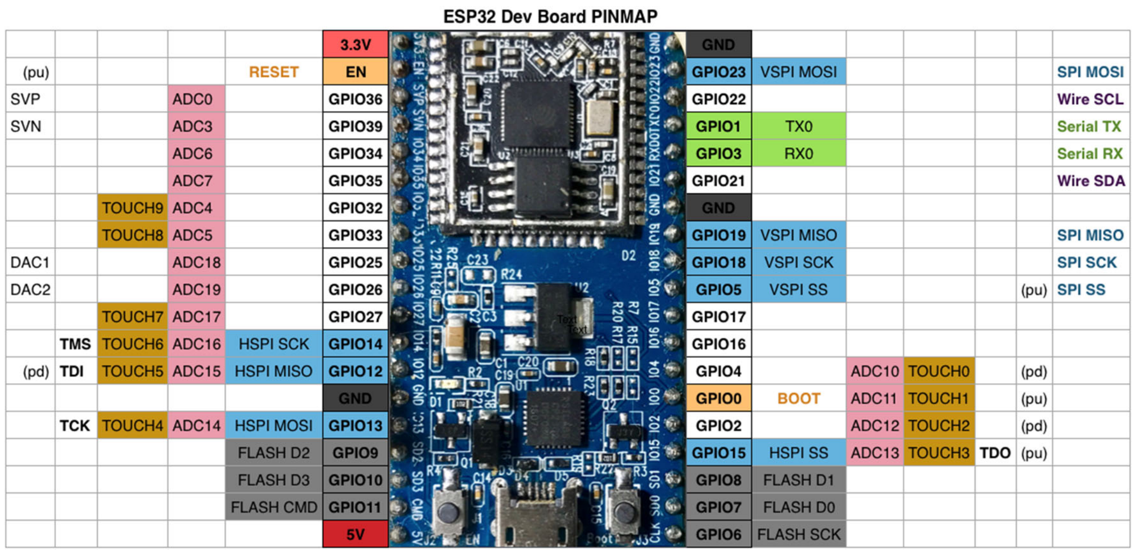
ESP32 workshop

For the class of 2022
Kajimoto Laboratory
ver. 1 2017.12.16
ver.2 2018.12.1
ver.3 2020 2.13
ver.4 2021 12.6
ver.5 2022

Change Log
Ver2:Bluetooth and Wi-Fi content change.
Ver.3: RC servo library change, DC motor control issue change, note about PWM, WiFi and multitasking are added.
Ver.4: Minor corrections
Ver.5: Encoder library relation correction.

1 Introduction

We will utilize ESP32-DevKitC (ESP32 is the name of the core chip). The library is updated daily, so there is always a possibility that the method in this document is out of date. The program on the PC side uses Processing in the course.



ESP32 Dev Board PINMAP

					3.3V		GND						
(pu)				RESET	EN		GPIO23	VSPI MOSI					SPI MOSI
SVP			ADC0		GPIO36		GPIO22						Wire SCL
SVN			ADC3		GPIO39		GPIO1	TX0					Serial TX
			ADC6		GPIO34		GPIO3	RX0					Serial RX
			ADC7		GPIO35		GPIO21						Wire SDA
		TOUCH9	ADC4		GPIO32		GND						
		TOUCH8	ADC5		GPIO33		GPIO19	VSPI MISO					SPI MISO
DAC1			ADC18		GPIO25		GPIO18	VSPI SCK					SPI SCK
DAC2			ADC19		GPIO26		GPIO5	VSPI SS				(pu)	SPI SS
		TOUCH7	ADC17		GPIO27		GPIO17						
	TMS	TOUCH6	ADC16	HSPI SCK	GPIO14		GPIO16						
(pd)	TDI	TOUCH5	ADC15	HSPI MISO	GPIO12		GPIO4		ADC10	TOUCH0		(pd)	
					GND		GPIO0	BOOT	ADC11	TOUCH1		(pu)	
	TCK	TOUCH4	ADC14	HSPI MOSI	GPIO13		GPIO2		ADC12	TOUCH2		(pd)	
				FLASH D2	GPIO9		GPIO15	HSPI SS	ADC13	TOUCH3	TDO	(pu)	
				FLASH D3	GPIO10		GPIO8	FLASH D1					
				FLASH CMD	GPIO11		GPIO7	FLASH D0					
					5V		GPIO6	FLASH SCK					

Figure. 1 ESP32-DevKitC Pin Assignment (https://github.com/espressif/arduino-esp32/blob/master/docs/esp32_pinmap.png)

Available at: https://akizukidenshi.com/catalog/g/gM-15673/
Breadboard:.http://akizukidenshi.com/catalog/g/gP-12366/ (6 rows instead of the

usual 5 rows. Because this kit is wider)

1.1 How to proceed with the assignment

- Chapter2 "Flashing LEDs": Flashing LEDs.
- Chapter3 "External switch": Input by external switch. Interrupt practice.
- Chapter 4 "Analog Input": Using analog input with the accelerometer as an example.
- Chapter 5 "Driving the RC Servo Motor": Drive the RC (radio-controlled) servo motor.
- Chapter6 "Analog output": Use the built-in DA converter.
- 7 Chapter 3 "SPI Communication": Expansion of DA conversion port as an example of port expansion by SPI communication.
- Chapter 8 "DC Motor Control": PD control of DC motor with H-bridge IC and encoder attached to the motor.
- Chapter9 "Data Logging": Practical data logging using Processing.
- Chapter10 "Wireless Communication": Running the Wi-Fi and Bluetooth communication samples.

1.2 report

Write a report at the end. Corrections will be made.

- The course is also a Microsoft-Word course, so please be sure to use the **reference** functions such as references to figure/table numbers. The Word training will be held in the middle of the workshop.
- Put up diagrams (especially oscilloscope screen captures) obtained during the experiment.
- **Do the experiment while writing the report**. This is the same in research situations. Document data in real time. Always take photos and videos. It is wrong to take a picture only when you write a paper.
- Submitted reports will be corrected and circulated throughout the lab.

2 LED Flashing

Set up the development environment and drive external LEDs.

- Using Arduino IDE, install ESP32 Dev Module library. Please search and check.

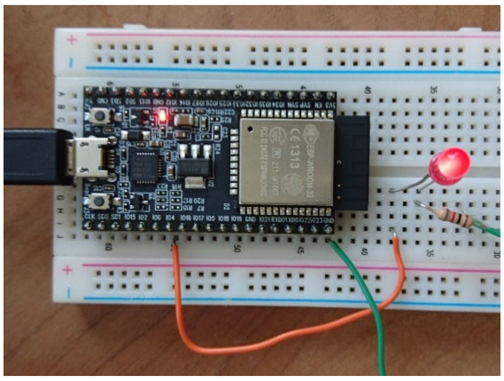
- Note the following points about the digital input/output of ESP32.
  - GPIO1 and 3 are used for serial communication.
  - GPIO6 to 11 are used internally, so their use is not recommended.

- ➢ GPIO34,35,36,39 are for input only
- ➢ As described later, GPIO0 doesn't go well with AD conversion.

- LED polarity should be checked with a tester.
- The resistor (limiting resistor) connected in series with the LED is set to 1k-OHM here, but it must be able to be calculated because the power supply voltage may vary. The calculation method of the resistance value is described in the following links, so learn it by yourself. If it is not a special one, 1 to 10mA flows roughly.
  http://www.ops.dti.ne.jp/~ishijima/sei/letselec/letselec11.htm
- The color code of the resistor should be readable.
- Breadboard is used here. When you remove it, use a flathead screwdriver etc. because it will be damaged if force is applied to the USB connector etc.
- The same net name in the schematic indicates that they are connected. For example, in the case of　Figure. 2 the two "GND" are connected.

[課題1]　Measure the electrical current through the LED (measure the voltage across the resistor and calculate it using Ohm's law). Is the value reasonable compared to the prediction obtained by the calculation method described above?

```
void setup() {
  pinMode(4, OUTPUT);
}

void loop() {
  digitalWrite(4, HIGH);
  delay(250);
  digitalWrite(4, LOW);
  delay(250);
}
```



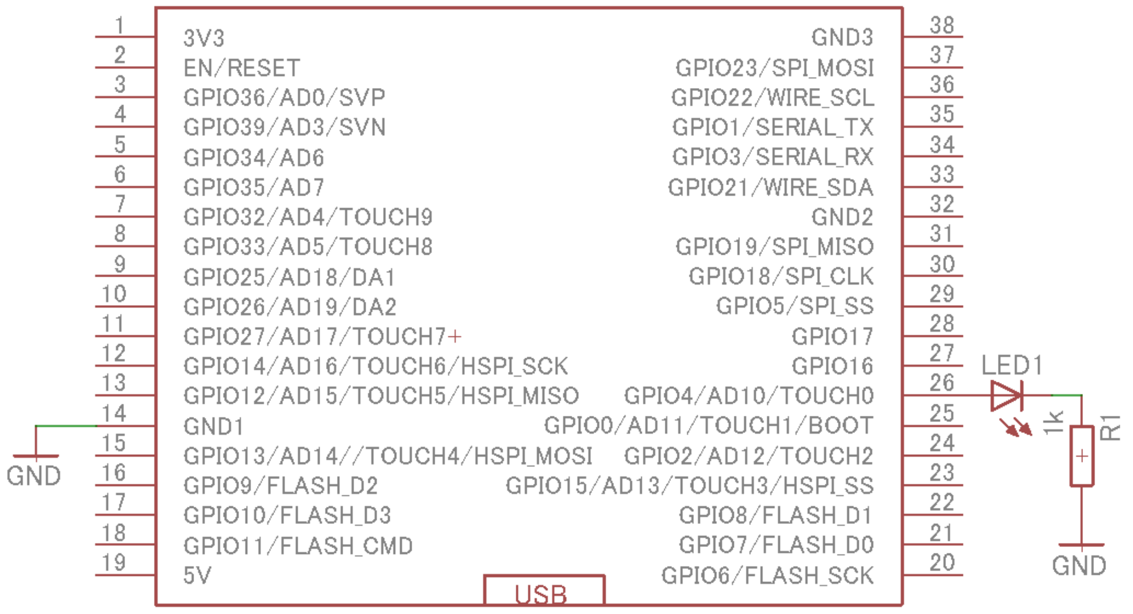


Figure. 2　LED Flashing Source and Circuit

3　External switch

External switch input is setup. (note that the ESP32 board also has an input switch). Refer to　Figure 3. 3 Check the conductivity of the tact switch with a tester when it is pressed.

[課題2]　Consider why a resistor is needed in a switch circuit. In the sample circuit shown below, what will be the logic input when the button is pressed?

[課題3]　Make the LED light up when the tact switch is pressed (use digitalRead function)

[課題4]　Learn about interrupt programming and modify your program to use interrupts. Make the LED blink every time the switch is pressed (use attachInterrupt function)

Reference http://gammon.com.au/interrupts

You should find that the blinking does not work perfectly as expected due to "chattering" of the switch. Google search "anti-chatter" and try the capacitor method or the software method.

[課題5]　The status of the switch is transmitted to PC by serial communication. For example, you can use serial communication software (RealTerm, TeraTerm, etc...) to send character "a" when the switch is pressed and "b" when the switch is not pressed. You can also use the monitor function of Arduino IDE.)

RealTerm http://realterm.sourceforge.net/

TeraTerm http://sourceforge.jp/projects/ttssh2/

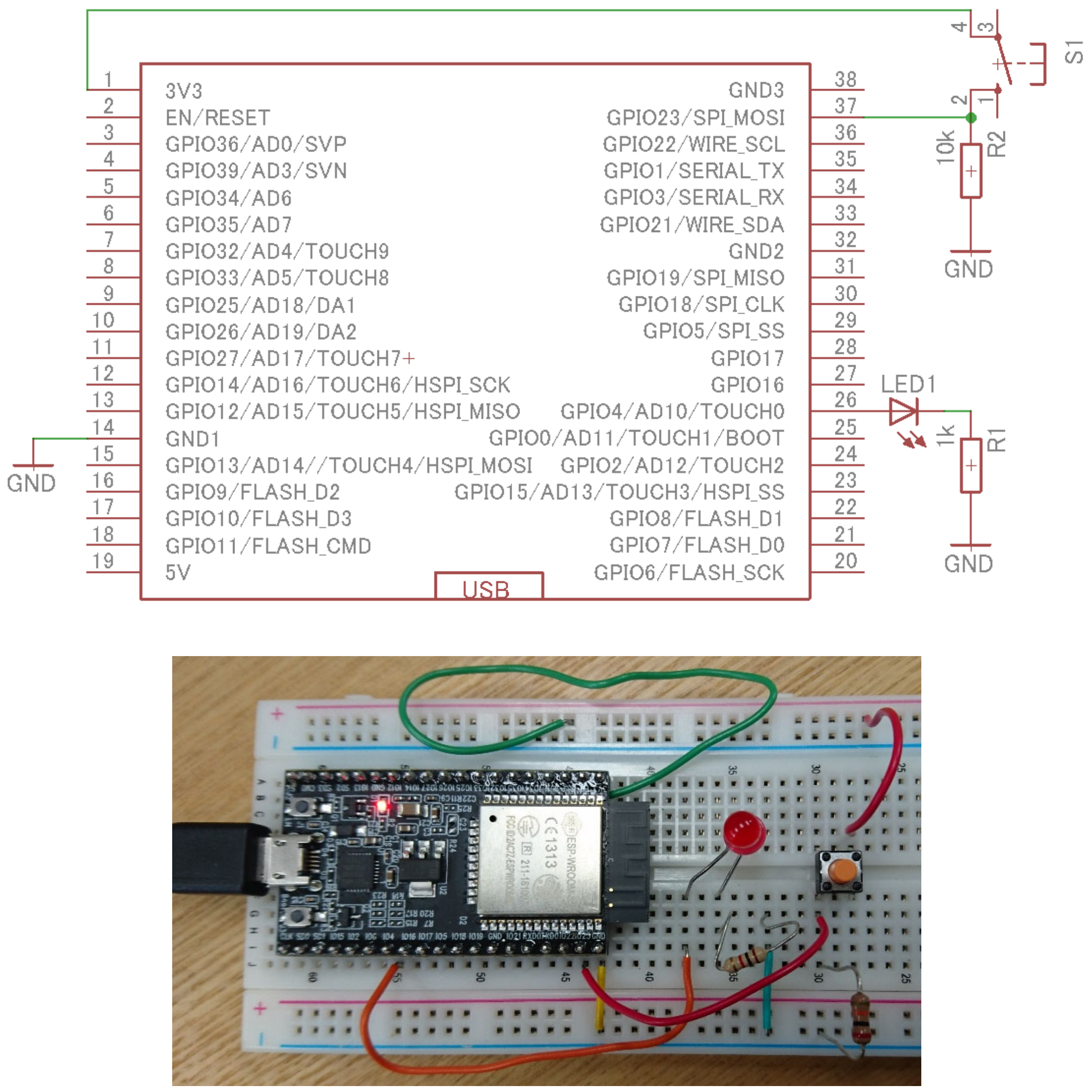


Figure 3. 3 Tact switch circuit part

4 Accelerometer

The KXR94-2050 module (http://akizukidenshi.com/catalog/g/gM-05153/) is used. This module provides analog output of 3-axis acceleration, so it should be received by 3 AD ports.

The function is analogRead, and the parameter of the function is not the channel number (0-19) of AD, but the GPIO number. For example, AD10 is not No. 10 but No. 4 (this is also true for DA, SPI, etc.).

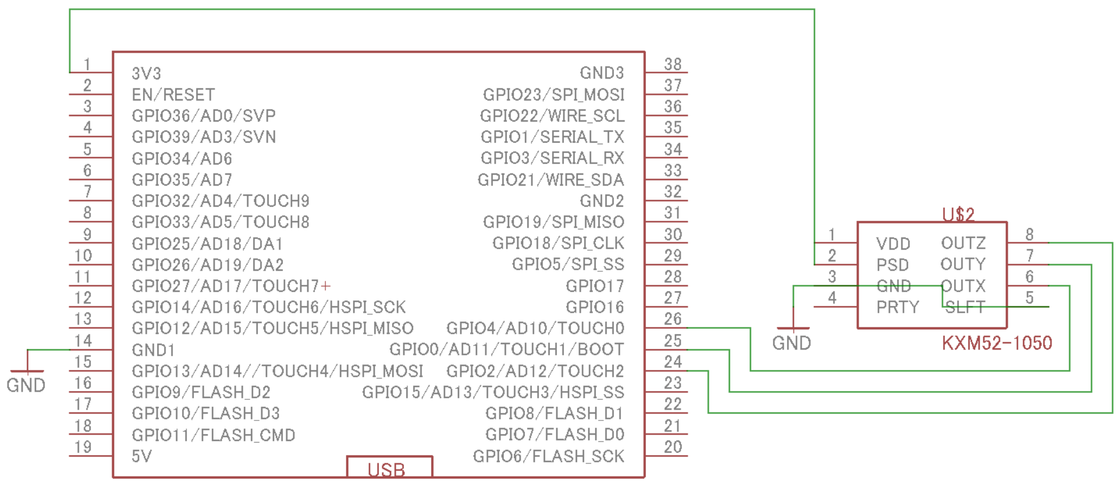
When this accelerometer is connected to 3.3V power supply, the offset is 1.65V and there is a change of 0.66V when 1G is applied. In other words, the output varies from 0.99V to 2.31V in the range of gravity acceleration. The AD conversion of ESP32 allocates 0 to 3.6V to 0 to 4095 (12bit) by default. Note that this is different from the normal Arduino output (10bit).

ESP32 has some limitations on AD conversion, for example, some channels cannot be used when using Wi-Fi function. Also, the attenuation function can change the measurement voltage range. http://trac.switch-science.com/wiki/esp32_tips

[課題6] Confirm that the values read by the PC are valid. Because of the existence of gravitational acceleration, the sensor value of an axis should be +1G in one direction and -1G in the opposite direction. What kind of value is actually changing? Is it reasonable?

(note: AD conversion of GPIO0 does not seem to go well. Use another port.

(http://esp-idf.readthedocs.io/en/latest/api-reference/peripherals/adc.html)



```
int ax,ay,az;

void setup() {
  Serial.begin(9600);
}

void loop() {
  ax = analogRead(4);
  ay = analogRead(0);
  az = analogRead(2);
  Serial.print(ax, DEC);  Serial.print(" ");
  Serial.print(ay, DEC);  Serial.print(" ");
  Serial.println(az, DEC);
  delay(500);
}
```

Figure. 4 Accelerometer circuit and source sample (note: GPIO0 is changed to another port. http://esp-idf.readthedocs.io/en/latest/api-reference/peripherals/adc.html)

5 RC servo motor drive (PWM)

PWM output is used to drive RC (radio-controlled) servo motors.

ESP32 is not soldered directly to the board, but a socket is used. The RC servo motors need high current to drive the servo motors and USB power supply is not enough.

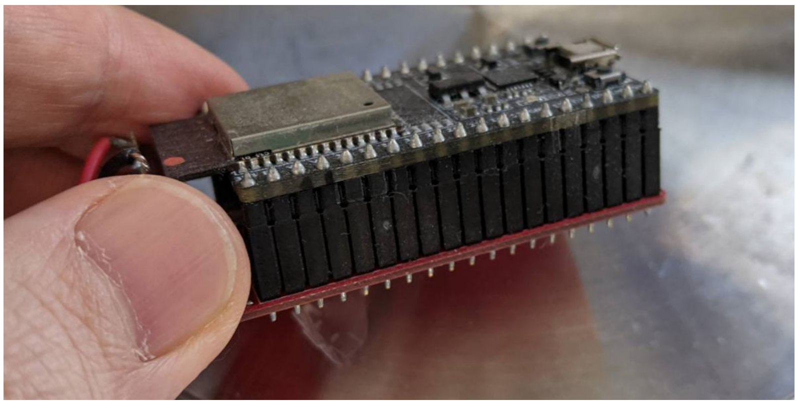


Figure. 5 Socket (this example is a separate project)

- Use tinned wire or junflon wire for wiring. Do not use unnecessarily thick wires. However, use thick wires for the part driving the motor and the power supply part.
  - Refer to the following link or "Elements of Electronics" for soldering method.
  - How to solder http://www.youtube.com/watch?v=S5f7jueQHr8
  - Good soldering geometry, etc. https://godhanda.co.jp/blog/kisokouza08/

Study control methods for PWM and RC servo motors

- http://berry.sakura.ne.jp/technics/servo_control_p1.html

- http://startelc.com/H8/H8_17RCserv.html

Use the RC servo library of ESP32.
https://wak-tech.com/archives/1534
https://github.com/RoboticsBrno/ServoESP32

Next, connect the RC servo motor to the power supply as shown in Figure 6 Connect the RC servo motor and the power supply as shown in Figure 6. In this circuit diagram, the 5V output of the AC adapter is connected to the 5V input terminal of ESP32. This makes it possible to operate without USB power supply. Do not forget to connect GND of AC adapter and GND of ESP32.

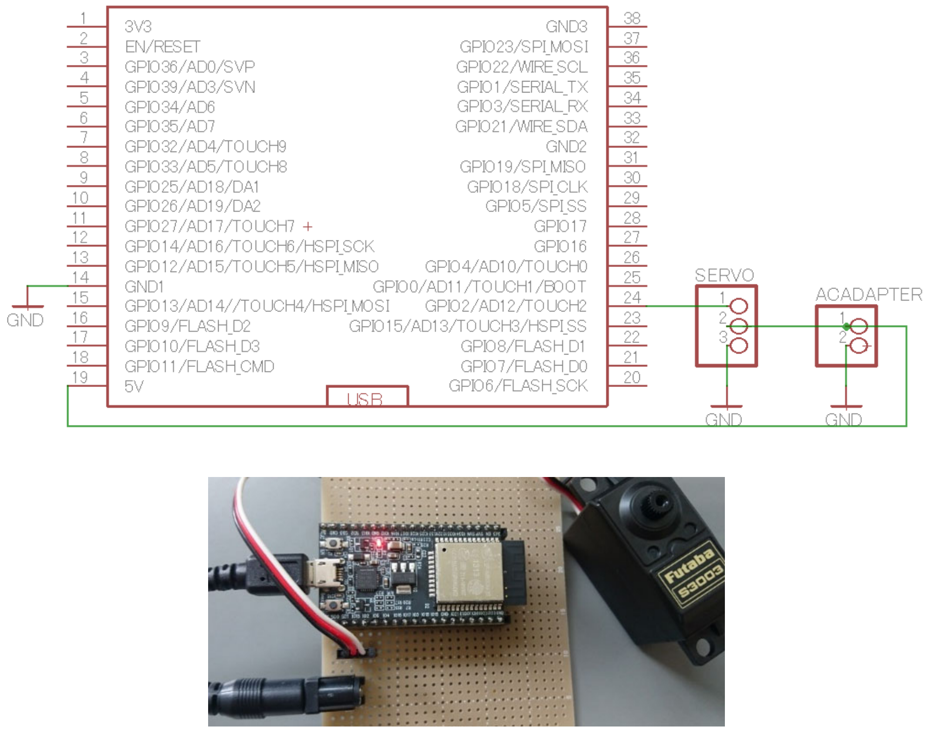


Figure 6 RC Servo Circuit

[課題7] Control the servo motor posture by serial communication from PC. For example, you can use serial communication software on PC side. Observe the PWM signal with an oscilloscope. (<u>"Observe with oscilloscope" means to capture the screen with oscilloscope and attach it to the report. For how to use the oscilloscope, refer to the following documents:</u> http://download.tek.com/document/55Z-17291-3.pdf http://download.tek.com/document/3GZ-24924-0.pdf You should know how to use Trigger function)

The Serial.available function is used to wait for input from the PC in serial communication. Figure. 7 Shows typical example.

```
void loop() {
  int incomingByte;

  if(Serial.available()>0){
    incomingByte = Serial.read();
    ここで送られてきたキーに応じた処理
  }
  delay(15);
}
```

Figure. 7 RC Servo and Serial Communication

6 Analog output (+ operational amplifier)

ESP32 has 2 channels of 8bit DA converter. The usual Arduino environment has analogWrite function as analog output, but <u>this is a pseudo analog output by PWM. On the other hand, ESP32 can actually convert DA output.</u>

[課題8] As shown in Figure. 8 the DA output is performed using the dacWrite function as shown in Figure 7, and the output waveform is checked with an oscilloscope. In this example, the output is from GPIO25 (DAC1). Consider what kind of waveform it is. Find the period and frequency of the waveform and the loop period.

```
int i=0;

void setup() {
}

void loop() {
  i=(i+1)%255;
  dacWrite(25,i);
}
```

Figure. 8 Analog output by dacWrite function

Driving earphones as an exercise in operational amplifier circuitry (Figure. 9). In this circuit the following is done.

① The DC component of the input signal from the DA is <u>cut off by the high-pass filter with </u>C1 and R1.

② It is amplified by the <u>inverting amplification circuit of </u>operational amplifier

(NJM4580 manufactured by New Japan Radio). The amplification ratio is specified by R1 and R4.

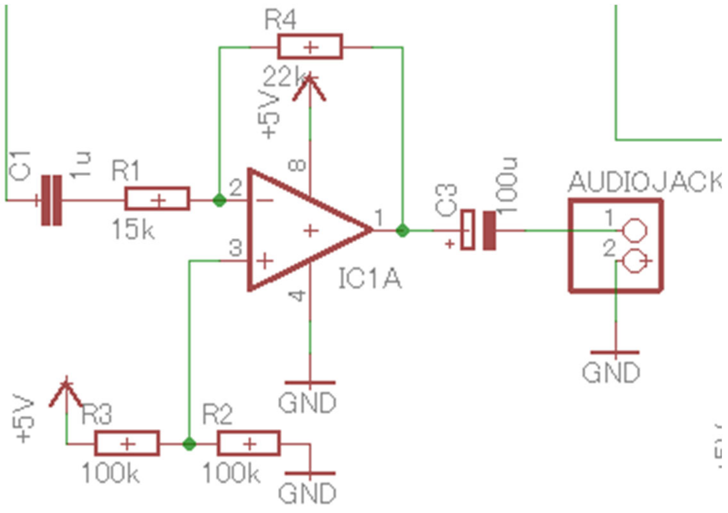
③ Offset voltage at amplification is set by R2 and R3 and input to V+ of op-amp.

④ The DC component of the output is cut by the high-pass filter by C3 and the resistance of the earphone.

⑤ Output from earphone jack to earphones or speakers.

- Note the polarity of the electrolytic capacitor. The + side should be the output side of the op-amp
- Use the type of earphone jack that can be directly attached to the board. In case of stereo jack, be careful about the polarity.
- You should learn how to read the numerical code for the capacitor.
 http://www.jarl.org/Japanese/7_Technical/lib1/konden.htm

Learn about operational amplifier circuits on your own in the following and other places.
http://picavr.uunyan.com/op_amp.html
https://www.analog.com/jp/education/landing-pages/003/opamp-application-handbook.html (Note: Too high level for this course)
https://kaji-lab.jp/ja/index.php?plugin=attach&pcmd=open&file=OpAmpIntro.pdf&refer=people%2Fkaji
The output from the ESP32 DAC can also be used to play earphones directly, and if that is the only purpose, an operational amplifier circuit is not necessary.



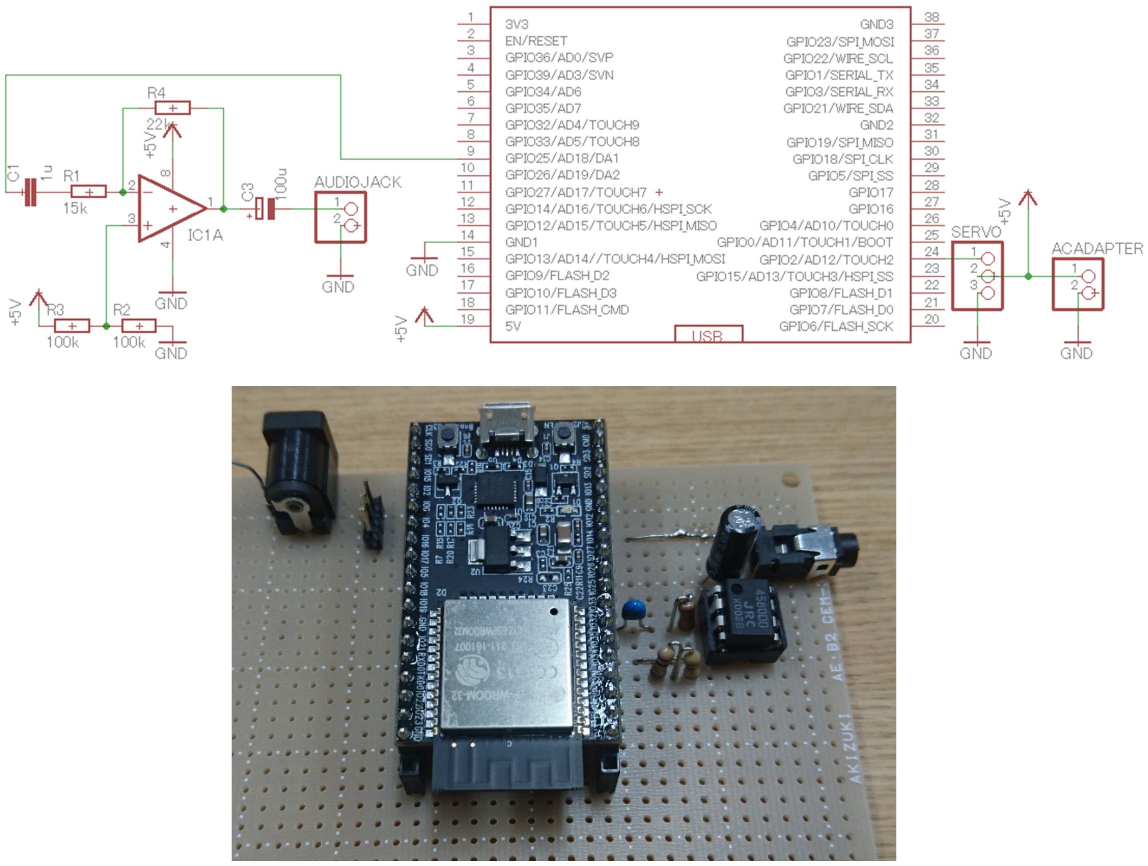


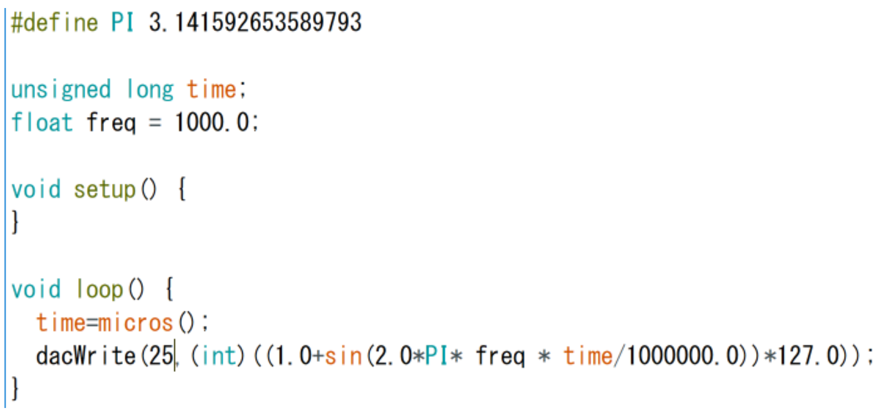
Figure. 9　Analog Output and Operational Amplifier Circuit

[課題9]　What is the amplification value of this inverting amplifier for the resistor values and capacitor capacitance? What is the expected voltage input from 0 to 3.3 V. What is the cutoff frequency due to the high-pass filter by C1 and R1 (e.g. http://sim.okawa-denshi.jp/CRhikeisan.htm). If the resistance of the earphone is 32-OHM, what is the cutoff frequency due to the high-pass filter by C3 and the earphone?

Note that in a typical audio circuit like this, the characteristics of the high-pass filter are determined only when a load such as earphones or speakers is connected. It is a typical error to measure without a load connected and the waveform is different from the expected one.

[課題10]　Create an actual circuit and listen to the input waveform (Do not use expensive earphones, as they may break the earphone!) Also, check the output waveform using an oscilloscope. The waveform should be different from the waveform of the DA port. How and why it is different (hint: positive/negative amplification factor, non-rail-to-rail)

[課題11] Control the sound of the earphone by serial communication from PC. When keys 1 to 8 are pressed, sound 1-8 is sent, sound "do-re-mi-fa-so-la-ti-do" is sounded. Here, **we use micros function to realize precise frequency wave.** The micros function returns the time in microseconds from the start of the program to the current time. For example, if you want to make a sound with a frequency of f[Hz], you should output a wave of sin(2 π ft). A sample of outputting a sine wave is shown in Figure. 10 Check the waveforms with an oscilloscope. If you observe it in detail, you should see a staircase waveform. Find the time taken for one cycle of the loop function from here.



```
#define PI 3.141592653589793

unsigned long time;
float freq = 1000.0;

void setup() {
}

void loop() {
  time=micros();
  dacWrite(25, (int)((1.0+sin(2.0*PI* freq * time/1000000.0))*127.0));
}
```

Figure. 10 Sinusoidal output using micros function

7 External DA (SPI communication for port extension)

ESP32 has several SPI communication modules. By using them, you can expand the ports. In this section, 8ch 10bit D/A port is realized.
Self-study on SPI communication in the following. It is assumed that you understand the shift register.
http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

The LTC1660CN 8-channel 10-bit D/A converter is used as an external DA converter (Figure 11). This exercise is also an exercise in reading the datasheet.
Put a capacitor for voltage stabilization between power supply and ground of DA converter. 10pin terminal is used and extra 2pin is 5V with ground.
http://akizukidenshi.com/catalog/g/gI-02794/

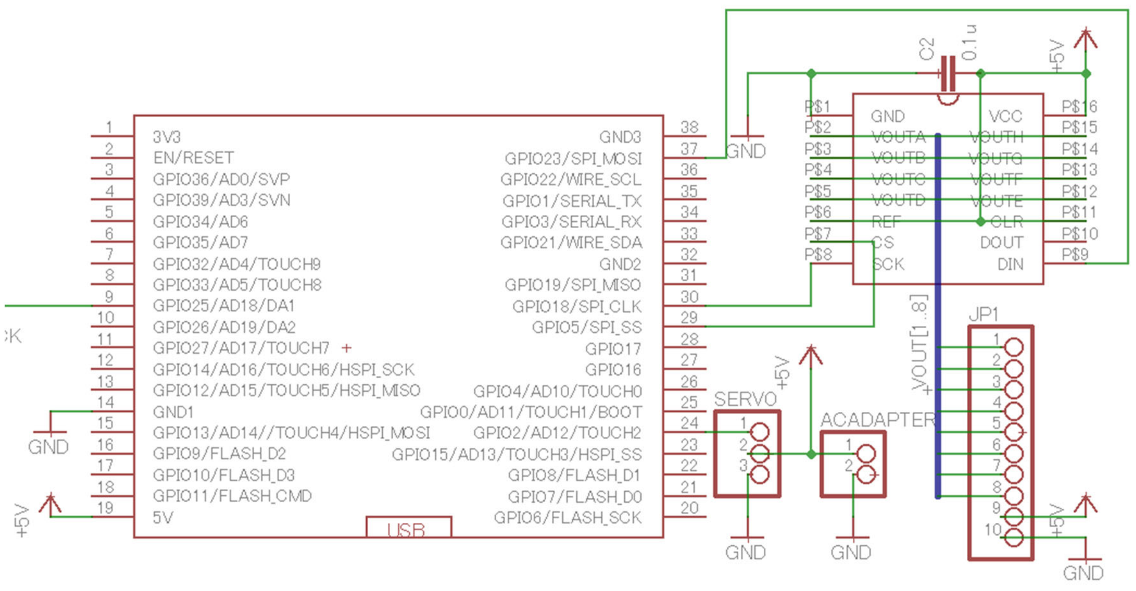
ESP32 has three kinds of SPI communication modules, SPI, VSPI and HSPI. SPI is used for communication with internal flash memory, and the Arduino environment uses one of them (VSPI, HSPI).

http://trac.switch-science.com/wiki/esp32_tips

https://www.mgo-tec.com/blog-entry-esp32-oled-ssd1331.html

https://www.mgo-tec.com/blog-entry-esp32-spimode-hspi-vspi-hispeed.html

https://github.com/espressif/arduino-esp32/commit/3198f25c19105ff933da4fdb33f132304bb3afaf



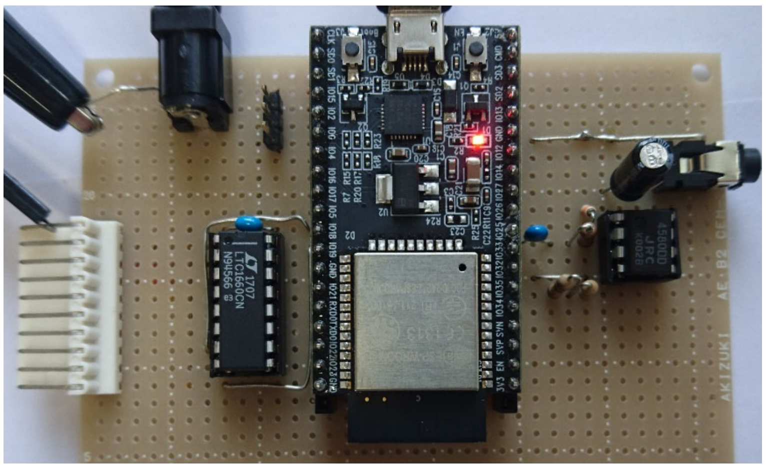


Figure 11 8ch DA port circuit (10 pins are used and the extra 2 pins are ground and 5V)

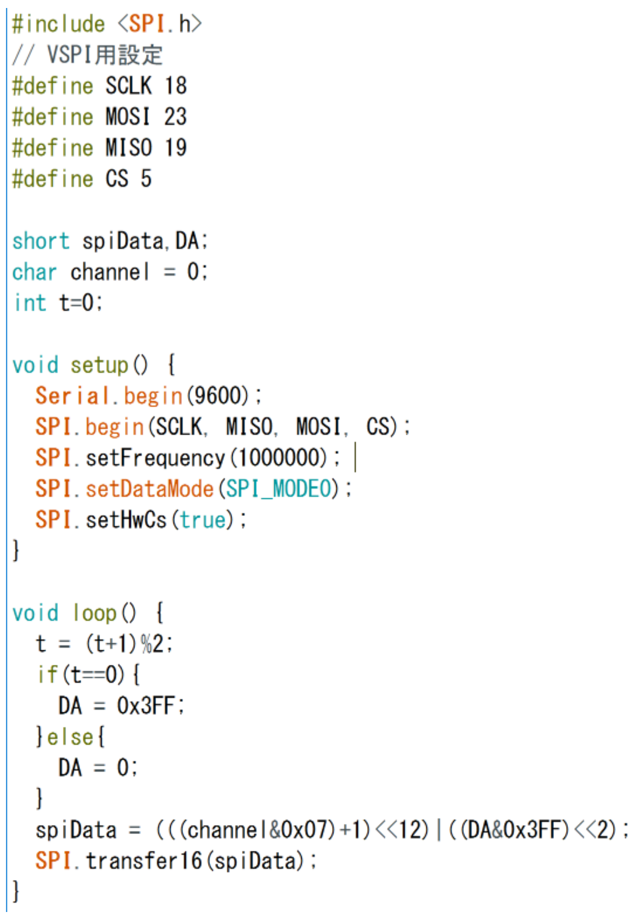
[課題12]　Run the following sample program and observe the waveforms using an oscilloscope. Next, observe the clock signal and the data signal at the same time (using the CS pin as the trigger signal. (The program uses two probes and the CS pin as the trigger signal.)

http://download.tek.com/document/55Z-17291-3.pdf

http://download.tek.com/document/3GZ-24924-0.pdf

What is the relationship between the timing of the two signals? Are they reasonable compared to the timing chart in the LTC1660 datasheet?

What is the relationship between the 16-bit data signals and the program? How do the 16-bit data signals relate to the program and are they appropriate compared to the LTC1660 datasheet?



```
#include <SPI.h>
// VSPI用設定
#define SCLK 18
#define MOSI 23
#define MISO 19
#define CS 5

short spiData,DA;
char channel = 0;
int t=0;

void setup() {
  Serial.begin(9600);
  SPI.begin(SCLK, MISO, MOSI, CS);
  SPI.setFrequency(1000000);
  SPI.setDataMode(SPI_MODE0);
  SPI.setHwCs(true);
}

void loop() {
  t = (t+1)%2;
  if(t==0){
    DA = 0x3FF;
  }else{
    DA = 0;
  }
  spiData = (((channel&0x07)+1)<<12)|((DA&0x3FF)<<2);
  SPI.transfer16(spiData);
}
```

Figure. 12 Sample of running 8ch D/A converter LTC1660.

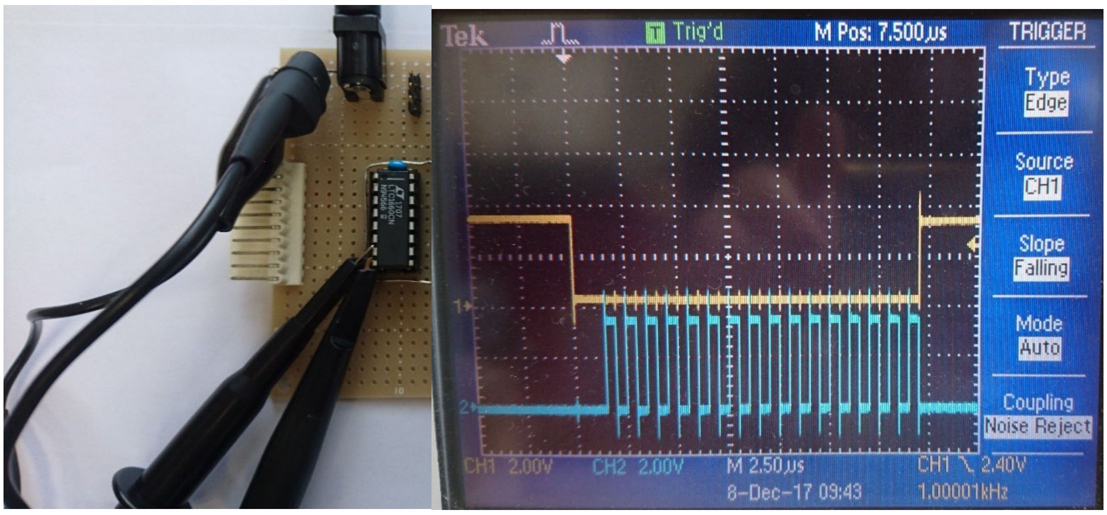


Figure. 13 Observation of control signal

[課題13] Create your own function that can output a voltage to a channel, such as void DAout(char channel, float voltage)

While looking at the LTC1660 datasheet, think about how to specify the channel. The voltage can be specified as a float from 0.0 to 5.0. Then, output sine waves of different frequencies from each of the 8 channels, and observe the waveforms with an oscilloscope. It takes time to calculate the sin function for 8 channels in real time, which may slow down the loop cycle. To solve this problem, create a table (array) of output data in the setup function in advance.

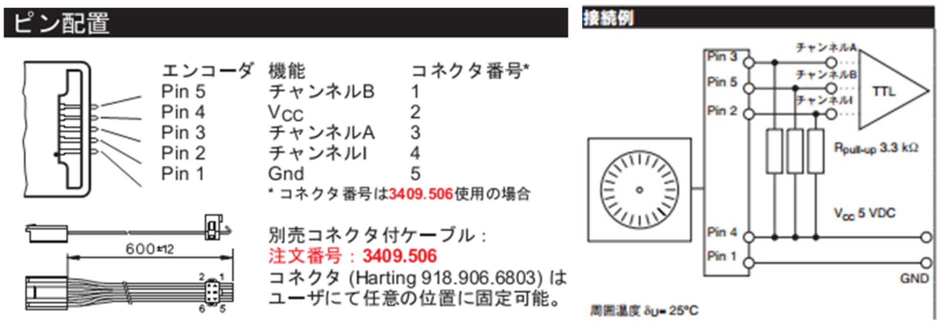
8 DC Motor Control

In this section, we drive a DC motor (RE25, 10W, 118746) manufactured by MAXON. The DC motor is equipped with an encoder (HEDS5540, 500 CPR (count per revolution)) to measure and control the position. For output, the H-bridge IC BD6222HFP is used.

Self-study encoders. Especially understand the quadruplicated mode.
http://ednjapan.com/edn/articles/1203/16/news012_2.html
https://www.sk-solution.co.jp/html/qa/qad001c00820090629163814.html
There is a large list of encoders in the Arduino environment below.
https://playground.arduino.cc/Main/RotaryEncoders

The following are used in this case.
https://www.arduino.cc/reference/en/libraries/esp32encoder/

Connect 5V, GND, and channels A and B as shown in Figure 13. Each output channel should be connected to the 5V power supply with a 3.3k ohm resistor (this is called a pull-up <u>resistor</u>: learn by yourself). It works without pull-up resistor, but it often causes trouble at high speed rotation.



ピン配置

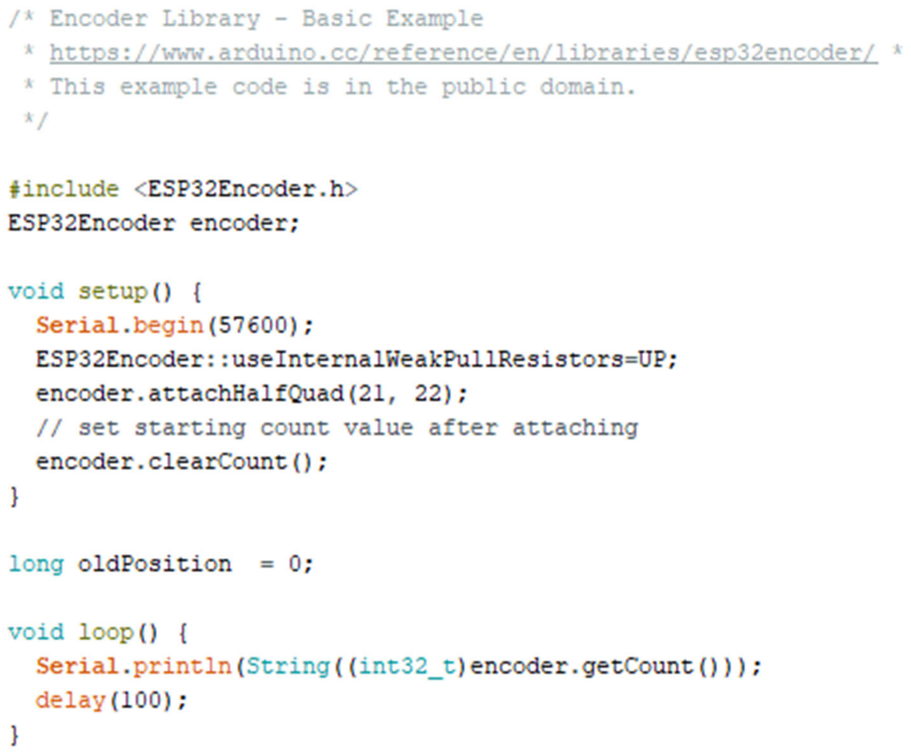
エンコーダ	機能	コネクタ番号*
Pin 5	チャンネルB	1
Pin 4	V_{CC}	2
Pin 3	チャンネルA	3
Pin 2	チャンネルI	4
Pin 1	Gnd	5

* コネクタ番号は3409.506使用の場合

別売コネクタ付ケーブル：
注文番号：3409.506
コネクタ (Harting 918.906.6803) は
ユーザにて任意の位置に固定可能。

Figure. 14 Encoder pinout (from HEDS5540 manual)

The source code is as follows (using GPIO pins 21 and 22). The encoder of 500 pulses per revolution is used in quadruplicated mode (how many pulses per revolution). The serial communication is set faster than before because of the frequency of the display.



```
/* Encoder Library - Basic Example
 * https://www.arduino.cc/reference/en/libraries/esp32encoder/ *
 * This example code is in the public domain.
 */

#include <ESP32Encoder.h>
ESP32Encoder encoder;

void setup() {
  Serial.begin(57600);
  ESP32Encoder::useInternalWeakPullResistors=UP;
  encoder.attachHalfQuad(21, 22);
  // set starting count value after attaching
  encoder.clearCount();
}

long oldPosition  = 0;

void loop() {
  Serial.println(String((int32_t)encoder.getCount()));
  delay(100);
}
```

Figure. 15　Sample program to read the encoder

[課題14]　Using an encoder, the angle of rotation of the motor is measured and continuously displayed on the PC screen via serial communication. The angle is converted to radian and displayed. Make a full revolution in the forward/reverse direction and check if a value of $\pm 2\pi$ is displayed.

(Reference) When the encoder value is measured by the microcomputer, counts are not taken when the rotation speed becomes high-speed, and the deviation may occur.

Study the H-bridge circuit by yourself, for example, by the following.
http://www.picfun.com/motor03.html

The motor drive (H-bridge drive) is controlled by PWM control. The BD6222HFP used in this project has 1.27mm pitch footprints, so it needs to be converted to normal board pitch (2.54mm) (Figure. 16). When using the pitch conversion board, make sure that the heat dissipation surface of the BD6222 does not contact each terminal (the heat dissipation surface itself is connected to the GND).

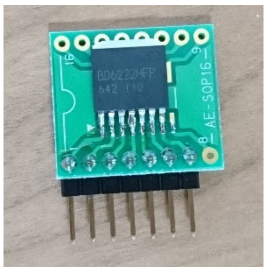


Figure. 16 Pitch conversion of BD6222HFP

The internal modules, pin names, and function table of the BD6222HFP are as shown in Figure. 17 By adding digital inputs to the Fin and Rin number pins, forward, reverse, brake, and stop are realized.
Put a 10uF capacitor between the power supply and GND. Select a capacitor with a withstand voltage of at least twice the power supply voltage.
As described later, the power supply for the motor drive should be separate from the 5V power supply of the ESP32.

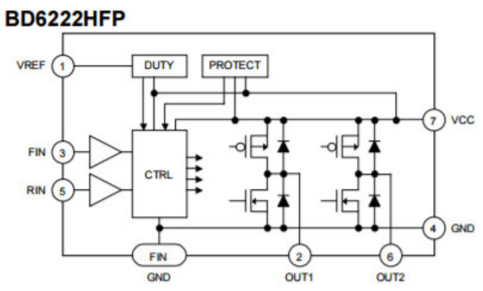
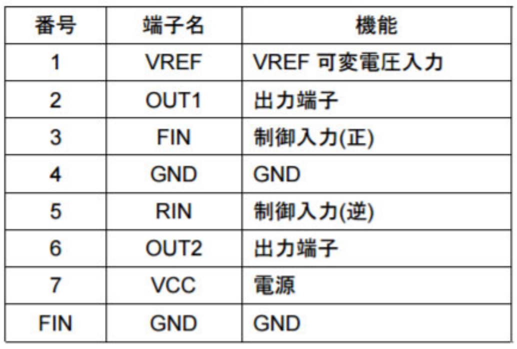


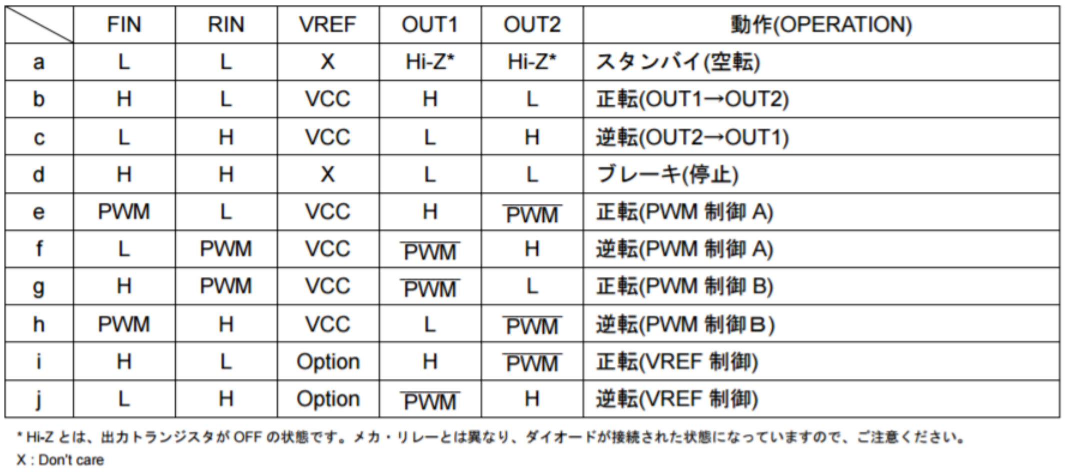
Fig 3　BD6222HFP

Table 2　BD6222HFP



番号	端子名	機能
1	VREF	VREF 可変電圧入力
2	OUT1	出力端子
3	FIN	制御入力(正)
4	GND	GND
5	RIN	制御入力(逆)
6	OUT2	出力端子
7	VCC	電源
FIN	GND	GND

Table 5　真理値表



	FIN	RIN	VREF	OUT1	OUT2	動作(OPERATION)
a	L	L	X	Hi-Z*	Hi-Z*	スタンバイ(空転)
b	H	L	VCC	H	L	正転(OUT1→OUT2)
c	L	H	VCC	L	H	逆転(OUT2→OUT1)
d	H	H	X	L	L	ブレーキ(停止)
e	PWM	L	VCC	H	$\overline{\text{PWM}}$	正転(PWM 制御 A)
f	L	PWM	VCC	$\overline{\text{PWM}}$	H	逆転(PWM 制御 A)
g	H	PWM	VCC	$\overline{\text{PWM}}$	L	正転(PWM 制御 B)
h	PWM	H	VCC	L	$\overline{\text{PWM}}$	逆転(PWM 制御B)
i	H	L	Option	H	$\overline{\text{PWM}}$	正転(VREF 制御)
j	L	H	Option	$\overline{\text{PWM}}$	H	逆転(VREF 制御)

*Hi-Z とは、出力トランジスタが OFF の状態です。メカ・リレーとは異なり、ダイオードが接続された状態になっていますので、ご注意ください。
X : Don't care

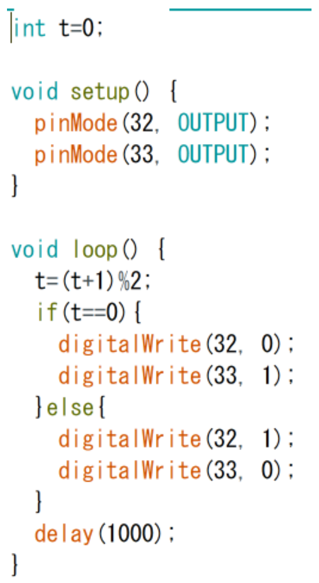
Figure. 17　Internal modules, pin names and truth table (from BD6222 manual)

The power supply for driving the motor is a regulated power supply for experimental use. This power supply has a voltage setting and current limit setting (CVCC power supply: Constant Voltage Constant Current power supply). In general, the power supply works as a constant voltage power supply when the current is less than the current limit, and becomes a constant current power supply when the current exceeds the limit. This allows the current limit to be set low enough to prevent circuit damage due to short circuits. Also, by actively using the current limit function, it may be used as a constant current source.

In this experiment, the power supply voltage is set to 12V and the current limit is set to about 0.3A at first. After that, if there is no problem on the circuit, the current limit is increased to about 1.5A.

Study the CVCC power supplies, for example, in the following.

http://www.kikusui.co.jp/knowledgeplaza/powersupply1/powersupply1_j.html
https://www.kikusui.co.jp/faq/topic.php?id=XOfJmMqF4OUx

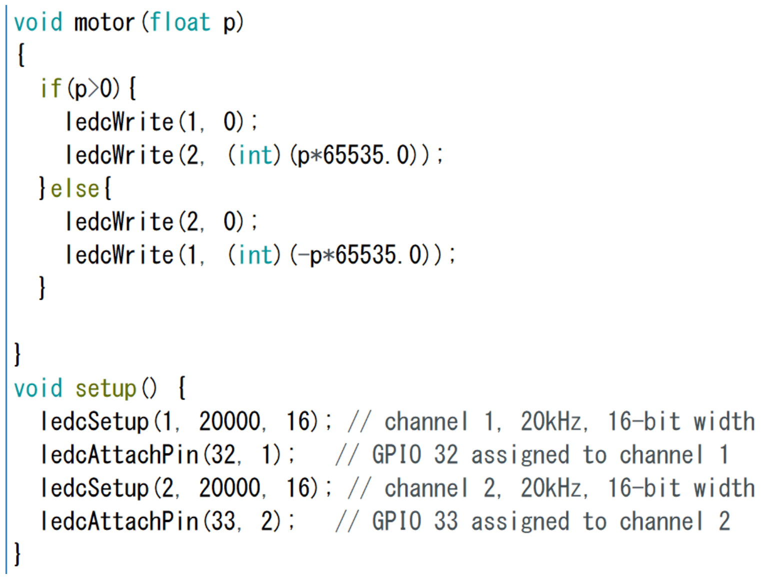


```
int t=0;

void setup() {
  pinMode(32, OUTPUT);
  pinMode(33, OUTPUT);
}

void loop() {
  t=(t+1)%2;
  if(t==0){
    digitalWrite(32, 0);
    digitalWrite(33, 1);
  }else{
    digitalWrite(32, 1);
    digitalWrite(33, 0);
  }
  delay(1000);
}
```

Figure 18　Sample program to run the motor driver

[課題15]　Explain the difference between a stop and a brake in an H-bridge circuit, including the principles.

The function table in Figure 18, together with the connection of PWM pins to Fin and Rin, suggests that the outputs can be controlled by a setup and function similar to Figure. 19 The output can be controlled by the setup and function as shown in Fig. 17.



```
void motor(float p)
{
  if(p>0){
    ledcWrite(1, 0);
    ledcWrite(2, (int)(p*65535.0));
  }else{
    ledcWrite(2, 0);
    ledcWrite(1, (int)(-p*65535.0));
  }

}
void setup() {
  ledcSetup(1, 20000, 16); // channel 1, 20kHz, 16-bit width
  ledcAttachPin(32, 1);   // GPIO 32 assigned to channel 1
  ledcSetup(2, 20000, 16); // channel 2, 20kHz, 16-bit width
  ledcAttachPin(33, 2);   // GPIO 33 assigned to channel 2
}
```

Figure. 19　Sample program to drive the H-bridge circuit with PWM (partial)

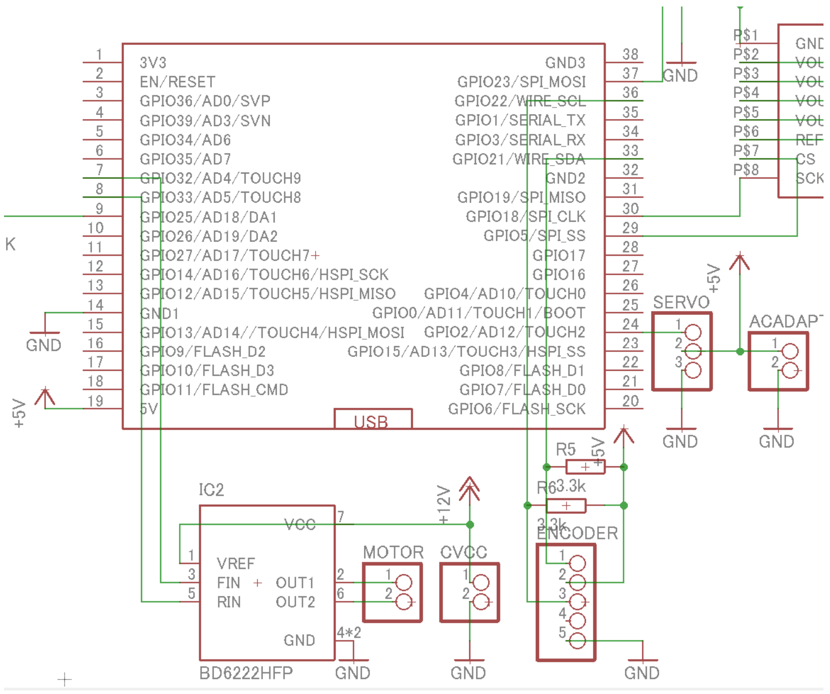


Figure. 20　Encoder reading circuit and H-bridge circuit

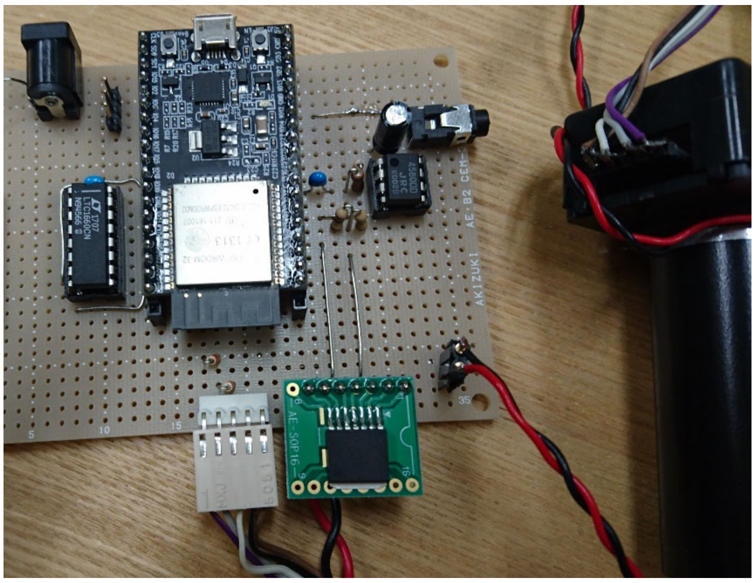


Figure. 21　Encoder Reading Circuit and H-Bridge Circuit Photo

[課題16]　Send commands via serial communication, for example, press f(forward) and b(backward) to switch between forward and reverse, and press h(igh) and l(ow) to change the speed each time. Set the frequency to 20kHz and check that the two PWM outputs are as expected by measuring two channels simultaneously with an oscilloscope. (Probably the motor is not driven at 20kHz. <u>This is related to the</u>

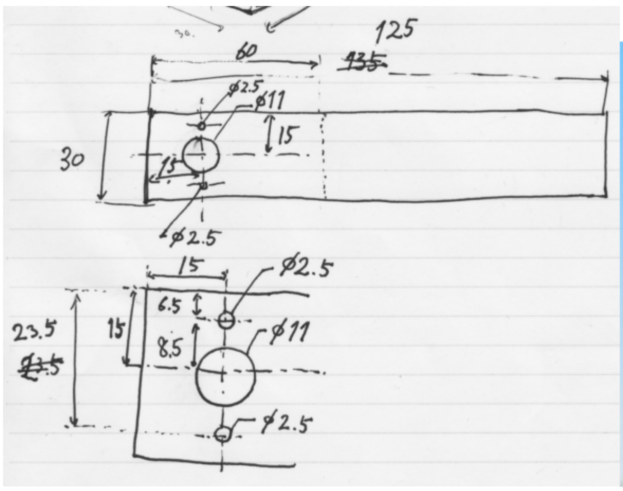
specified bit width. Change the bit width to 12bit and use 20kHz setting.

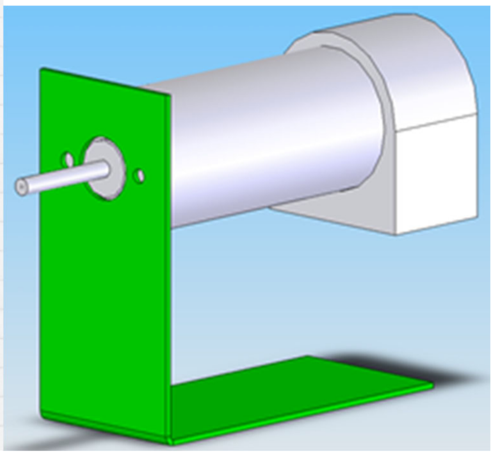
[課題17] How much current is flowing when the motor is kept rotating in one direction? Also, when you stop the rotation of this motor by hand (be careful!), how much current flows? Explain why this difference occurs (keywords: DC motor, back EMF).

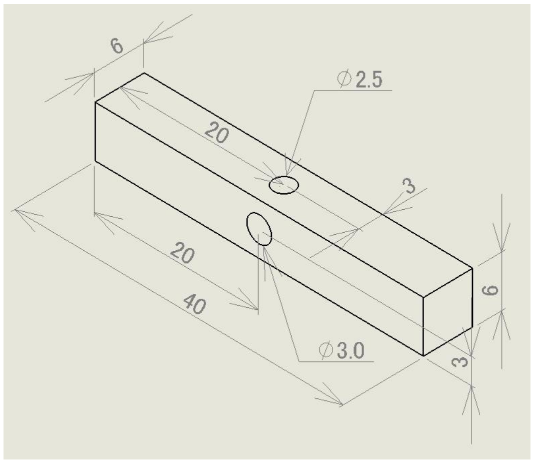
[課題18] Attach a handle to the DC motor shaft by machining aluminum. (This part will be taught individually according to the following)

Aluminum machining course content

1. How to take care of your teeth
   ① How to Use Calipers
   ② How to use the Height Gauge
   ③ How to punch
2. How to use a band saw
3. How to use the drill
   ① How to loosen the pedestal and move it up and down
   ② How to replace the drill
   ③ Fixing materials in a vice
   ④ Drilling, deburring for small holes
   ⑤ Special Drill 1: How to use the Step Drill
   ⑥ Special Drill 2: How to Use Deburring Drills
4. How to cut the tap
5. Clean up: How to clean up (Brush around tools. Vacuum the floor)







From now on, fix the aluminum base to the desk with duct tape (Figure. 22).

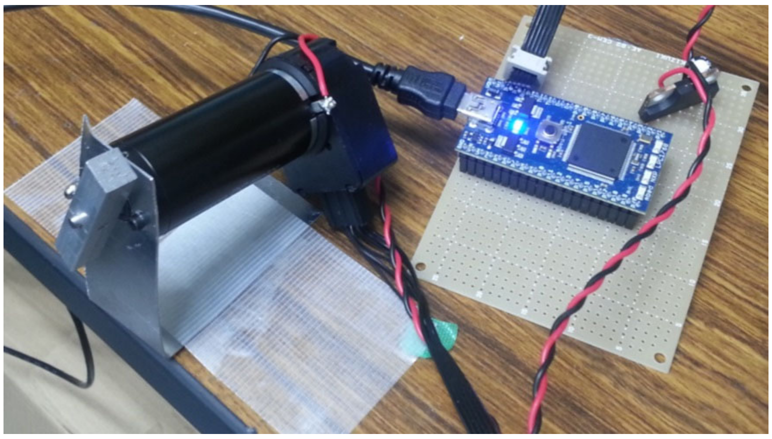


Figure. 22　Knobbed motor and control board

[課題19]　　Use the P control to head to the desired position (the initial position is fine), turn on the power and try to rotate the handle by hand. See what happens. Increase the gain and try to return as quickly as possible.

The loop cycle should be measured so that the 1kHz loop is always maintained. You can use the timer interrupt described later, or you can insert an appropriate delay in the loop function.

[課題20]　　Try to get the PD control to go to the desired position (initial position is fine), turn on the power, and rotate the handle by hand. What happens? Also, send position commands from PC (e.g. r(ight), l(eft)) and make it move every time you press the keyboard. In order to calculate the D component, the derivative is necessary, but be sure to measure the loop period and calculate it using the time ⊿t (to cope with changes in the loop period).

[課題21]　　The target trajectory is drawn by PID control. For example, one second rotation is used as the orbit. The PID control is applied to the difference between the current position and the target trajectory. Send commands from the PC (e.g. r(ight), l(eft)) so that the direction of rotation changes each time the keyboard is pressed.

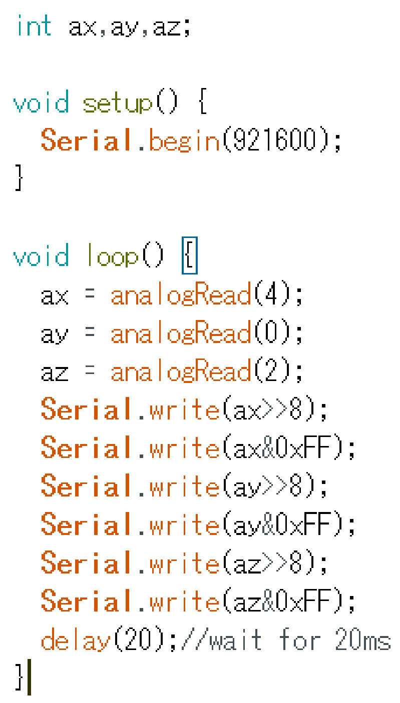
[課題22]　　Represent a "wall". In this case, we can express a spring wall by considering the angle beyond a certain angle as the "amount of bending" and presenting a reaction force proportional to this amount of bending. If the proportional constant (spring constant) is increased, a harder wall can be expressed. The proportional constant should be as high as possible. The loop period is set to 100Hz, 1kHz, and 10kHz, and how the hardness changes is observed.

9 Data logging and Processing by PC program

We practiced how to read the values of three axes of acceleration sensor in chapter 2. In this chapter, we consider the data logging method using PC program.

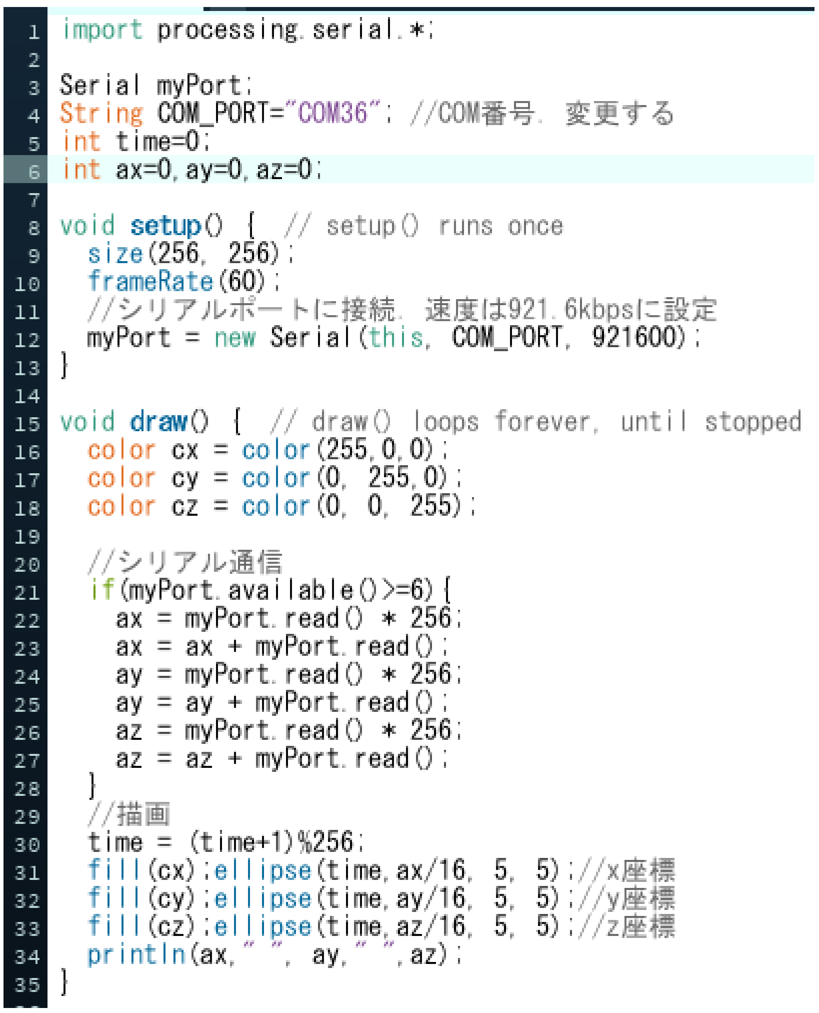
From here on, we will use Processing as the programming environment on the PC side (https://processing.org/). If you have never used it before, you should learn it by yourself before proceeding to the following tasks. The circuit in Section 2.2 is created on the board before proceeding. The power supply to the accelerometer is the 3.3V power supply of ESP32.

[課題23]　　The sample program is shown in Figure 24 Modify the Processing program and save the data in CSV file format. Use PrintWriter class, see the help of Processing. Also check the keyboard related functions, and start recording when any key is pressed, and save the data after 100 recordings and exit.



```
int ax,ay,az;

void setup() {
  Serial.begin(921600);
}

void loop() {
  ax = analogRead(4);
  ay = analogRead(0);
  az = analogRead(2);
  Serial.write(ax>>8);
  Serial.write(ax&0xFF);
  Serial.write(ay>>8);
  Serial.write(ay&0xFF);
  Serial.write(az>>8);
  Serial.write(az&0xFF);
  delay(20);//wait for 20ms
}
```

Figure 23　Accelerometer sample program (ESP32 side)



```
import processing.serial.*;

Serial myPort;
String COM_PORT="COM36"; //COM番号．変更する
int time=0;
int ax=0,ay=0,az=0;

void setup() {  // setup() runs once
  size(256, 256);
  frameRate(60);
  //シリアルポートに接続．速度は921.6kbpsに設定
  myPort = new Serial(this, COM_PORT, 921600);
}

void draw() {  // draw() loops forever, until stopped
  color cx = color(255,0,0);
  color cy = color(0, 255,0);
  color cz = color(0, 0, 255);

  //シリアル通信
  if(myPort.available()>=6){
    ax = myPort.read() * 256;
    ax = ax + myPort.read();
    ay = myPort.read() * 256;
    ay = ay + myPort.read();
    az = myPort.read() * 256;
    az = az + myPort.read();
  }
  //描画
  time = (time+1)%256;
  fill(cx);ellipse(time,ax/16, 5, 5);//x座標
  fill(cy);ellipse(time,ay/16, 5, 5);//y座標
  fill(cz);ellipse(time,az/16, 5, 5);//z座標
  println(ax," ", ay," ",az);
}
```

Figure 24　Sample program for reading acceleration sensor value (Processing side)

In the above sample, serial communication was performed in the Draw function in Processing. However, the Draw function is usually read at 60fps. There is a limitation if you want to record data more frequently or accurately.

Processing's serial library has a mechanism that is useful in this case. First, define the serialEvent function. Also use the bufferUntil function to make sure that a serialEvent is fired only when a specific character is encountered. To use this mechanism, you need to specify a specific character at the end of the data (called footer from now on), so the amount of sending/receiving will increase by 1 byte per time (Figure. 25).

At this time, it is <u>necessary to avoid that the numerical value indicating the footer appears in the data by accident</u>. Since the AD conversion is 12 bits per channel this time, if we transmit the upper and lower 6 bits each, all the data will be within the range of 0-63. If the numerical value of the footer is, for example, 0xFF (all 8 bits are 1), there is no need to worry about the conflict of the data and the specified character.

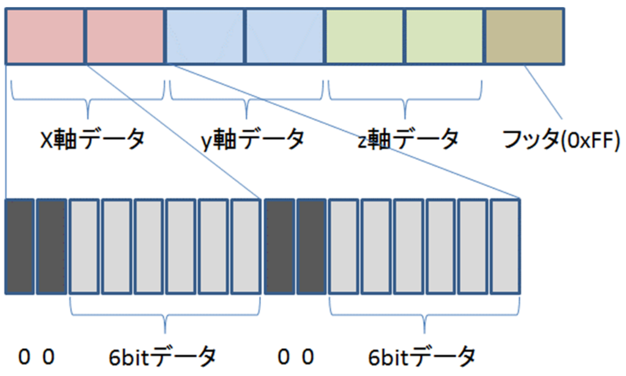
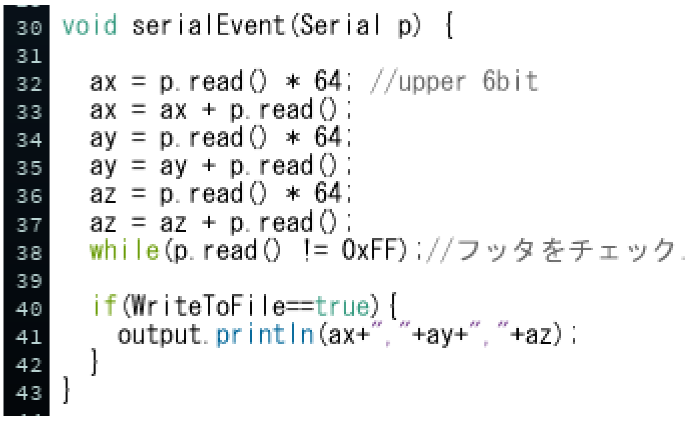


Figure. 25　Example of data column and footer

When reading multiple bytes of data, it is important to ensure that the "break" in the data is correct. The data from the microcontroller may start from the Y-axis data or the second byte of the Z-axis data, depending on the timing of starting the microcontroller, the timing of starting the program on the PC, and so on.

To solve this problem, the first method is to send a request signal for data transmission from the PC side to the mbed side. This is the most standard way, but there is also a way to explicitly search the footer in the program on the PC side, as in　Figure. 26 However, there is also a way to explicitly search the footer in the program on the PC side, as shown in Fig. 26. This way, the break of data is correctly recognized at the first read.



```
void serialEvent(Serial p)  {

  ax = p.read() * 64; //upper 6bit
  ax = ax + p.read();
  ay = p.read() * 64;
  ay = ay + p.read();
  az = p.read() * 64;
  az = az + p.read();
  while(p.read() != 0xFF);//フッタをチェック.

  if(WriteToFile==true){
    output.println(ax+","+ay+","+az);
  }
}
```

Figure. 26　Searching for a footer in a serial event (Processing)

The definition of interrupt functions with bufferUntil can lead to non-reproducible bugs, as is usual with such interrupt functions, because it is not known when the serialEvent function will be called. For example, the serialEvent function may be called the moment after bufferUntil is called (before Processing's setup function has finished). This can cause bugs, for example, if the file to be written is not yet ready to be written. Such bugs typically occur "once in a while". Do not use ad-hoc solutions to these bugs.

[課題24]　Record the values of 3 axes of acceleration sensor at 1kHz. To make the interval of data acquisition on ESP32 side precise, perform timer interrupt by referring to the above explanation To make the interval of data acquisition on ESP32 side accurate, perform timer interrupt with referring to the above explanation. In order to confirm that the measurement is done in 1ms cycle, invert the state of appropriate digital output pin every time and observe it with oscilloscope.

ESP32 can use timer interrupt with high time resolution.
http://www.iotsharing.com/2017/06/how-to-use-interrupt-timer-in-arduino-esp32.html
https://esp32.com/viewtopic.php?f=19&t=2481
https://techtutorialsx.com/2017/10/07/esp32-arduino-timer-interrupts/
https://github.com/espressif/arduino-esp32/blob/master/libraries/ESP32/examples/Timer/RepeatTimer/RepeatTimer.ino

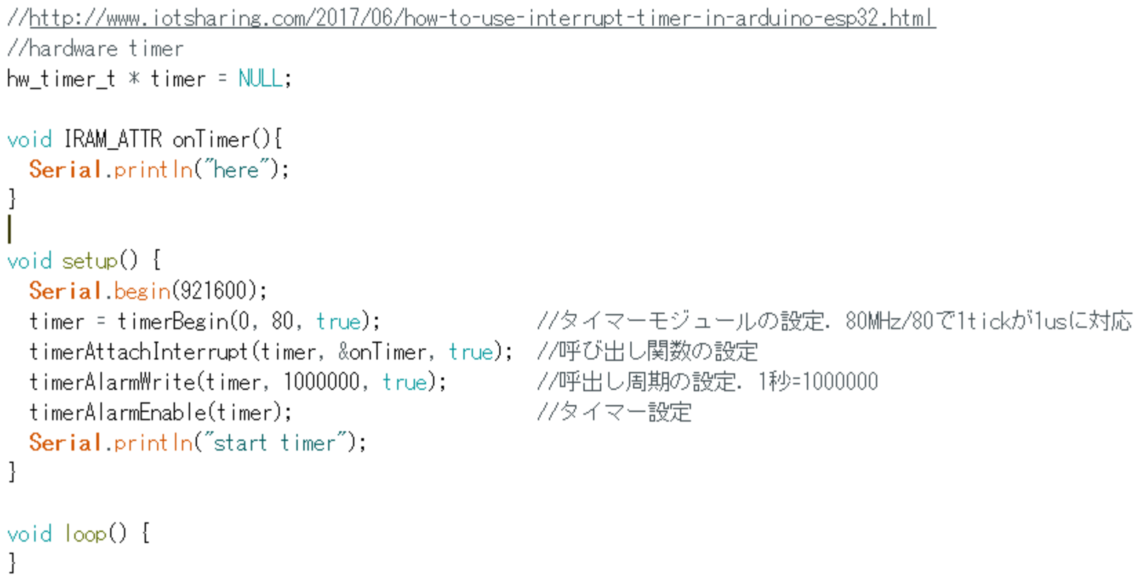

```
//http://www.iotsharing.com/2017/06/how-to-use-interrupt-timer-in-arduino-esp32.html
//hardware timer
hw_timer_t * timer = NULL;

void IRAM_ATTR onTimer(){
  Serial.println("here");
}

void setup() {
  Serial.begin(921600);
  timer = timerBegin(0, 80, true);              //タイマーモジュールの設定. 80MHz/80で1tickが1usに対応
  timerAttachInterrupt(timer, &onTimer, true);  //呼び出し関数の設定
  timerAlarmWrite(timer, 1000000, true);        //呼出し周期の設定. 1秒=1000000
  timerAlarmEnable(timer);                      //タイマー設定
  Serial.println("start timer");
}

void loop() {
}
```

Figure. 27　Timer interrupt sample program

The above is a method of measurement that does not depend on the cycle of the Draw function on the Processing side. However, the stability of the USB communication is a problem when the control loop including a PC is configured. The high-speed control loop should be completed in the microcontroller whenever possible. If real-time performance is not required, temporary storage in the memory on the microcontroller is often used.

[課題25]　Control using a DC motor and an acceleration sensor. The acceleration sensor is attached to an aluminum rod at the end of the DC motor so that the aluminum rod is always horizontal even when the DC motor is moved by hand. The measurement and control loop is set to 1 kHz or higher using timer interrupt.

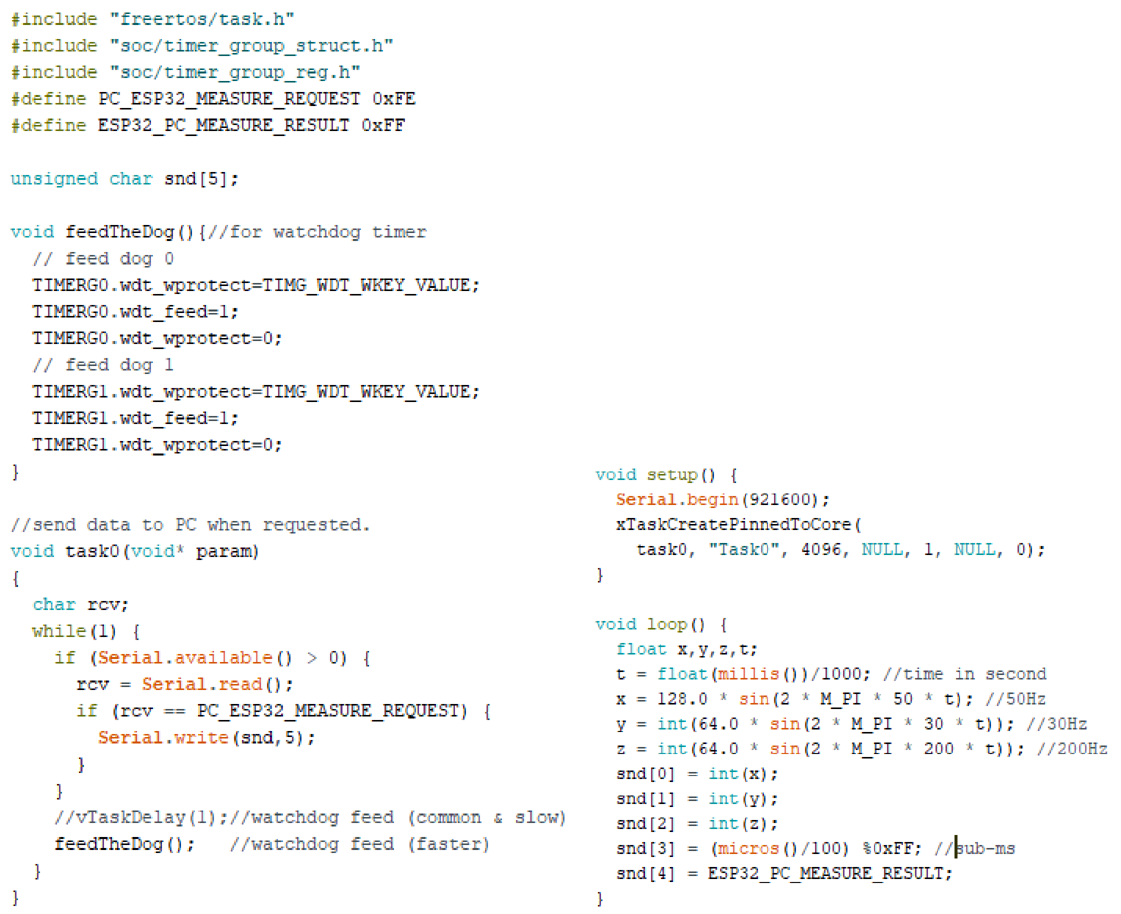
9.1 Multitasking

The example above was to set the control loop to 1kHz by timer interrupt of ESP32. On the other hand, if the ESP32 has more tasks to do or more data to send, the communication overhead becomes a problem (e.g. when using hundreds of sensor data points).

In such a case, ESP32 has two CPU cores and can be stabilized by dividing it into "core for serial communication" and "core for measurement and control". For example, measurement and control is done at 1kHz, and serial communication is used to receive commands from PC or to send the current status to PC (1kHz is also good for this, but a lower cycle is also acceptable depending on the purpose).

https://plaza.rakuten.co.jp/lovesun/diary/201908100000/

https://forum.arduino.cc/index.php?topic=621311.02

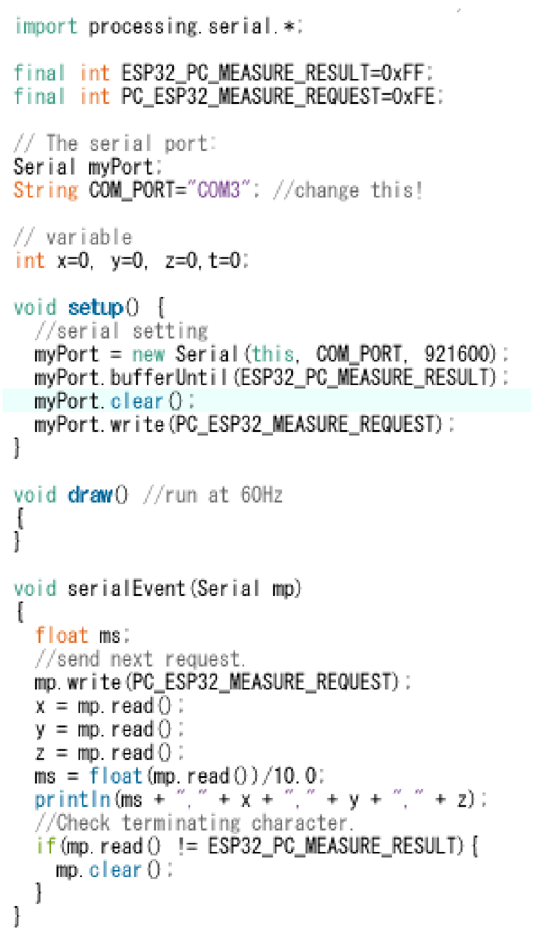


```
#include "freertos/task.h"
#include "soc/timer_group_struct.h"
#include "soc/timer_group_reg.h"
#define PC_ESP32_MEASURE_REQUEST 0xFE
#define ESP32_PC_MEASURE_RESULT 0xFF

unsigned char snd[5];

void feedTheDog(){//for watchdog timer
  // feed dog 0
  TIMERG0.wdt_wprotect=TIMG_WDT_WKEY_VALUE;
  TIMERG0.wdt_feed=1;
  TIMERG0.wdt_wprotect=0;
  // feed dog 1
  TIMERG1.wdt_wprotect=TIMG_WDT_WKEY_VALUE;
  TIMERG1.wdt_feed=1;
  TIMERG1.wdt_wprotect=0;
}

//send data to PC when requested.
void task0(void* param)
{
  char rcv;
  while(1) {
    if (Serial.available() > 0) {
      rcv = Serial.read();
      if (rcv == PC_ESP32_MEASURE_REQUEST) {
        Serial.write(snd,5);
      }
    }
    //vTaskDelay(1);//watchdog feed (common & slow)
    feedTheDog();   //watchdog feed (faster)
  }
}
```

```
void setup() {
  Serial.begin(921600);
  xTaskCreatePinnedToCore(
    task0, "Task0", 4096, NULL, 1, NULL, 0);
}

void loop() {
  float x,y,z,t;
  t = float(millis())/1000; //time in second
  x = 128.0 * sin(2 * M_PI * 50 * t); //50Hz
  y = int(64.0 * sin(2 * M_PI * 30 * t)); //30Hz
  z = int(64.0 * sin(2 * M_PI * 200 * t)); //200Hz
  snd[0] = int(x);
  snd[1] = int(y);
  snd[2] = int(z);
  snd[3] = (micros()/100) %0xFF; //sub-ms
  snd[4] = ESP32_PC_MEASURE_RESULT;
}
```

Figure. 28 Multitasking sample program (ESP32)

The normal loop() function is running on core1, so it is processed in parallel on core0 (in this case, serial communication is used. This example doesn't have the advantage of multitasking. In the parallel processing side, it is often used to prevent Watchdog Timer-related stoppage by calling vTaskDelay() function. In the above example, we define a function called feedTheDog instead of this function to explicitly prevent the Watchdog Timer from stopping.
Multitasking is not a special assignment, so please refer to it when you need it in a real assignment. 2 cores are included in ESP32.



```
import processing.serial.*;

final int ESP32_PC_MEASURE_RESULT=0xFF;
final int PC_ESP32_MEASURE_REQUEST=0xFE;

// The serial port:
Serial myPort;
String COM_PORT="COM3"; //change this!

// variable
int x=0, y=0, z=0,t=0;

void setup() {
  //serial setting
  myPort = new Serial(this, COM_PORT, 921600);
  myPort.bufferUntil(ESP32_PC_MEASURE_RESULT);
  myPort.clear();
  myPort.write(PC_ESP32_MEASURE_REQUEST);
}

void draw() //run at 60Hz
{
}

void serialEvent(Serial mp)
{
  float ms;
  //send next request.
  mp.write(PC_ESP32_MEASURE_REQUEST);
  x = mp.read();
  y = mp.read();
  z = mp.read();
  ms = float(mp.read())/10.0;
  println(ms + "," + x + "," + y + "," + z);
  //Check terminating character.
  if(mp.read() != ESP32_PC_MEASURE_RESULT){
    mp.clear();
  }
}
```

Figure. 29 Corresponding sample program (Processing)

10 Wireless communication

The board used in this project has modules for Wi-Fi and Bluetooth. Use them!

10.1 BluetoothSerial

BluetoothSerial library is used. No download is necessary if you have the latest version of Arduino-ESP32 environment.

https://github.com/don/BluetoothSerial

https://blog.rogiken.org/blog/2018/04/11/esp32%E3%81%A7bluetoothserial%E9%80%9A%E4%BF%A1%E3%82%92%E3%81%97%E3%82%88%E3%81%86%EF%BC%81/

https://techtutorialsx.com/2018/03/13/esp32-arduino-bluetooth-over-serial-receiving-data/

If you add Bluetooth device on PC side, serial port will be increased, so you can write normal serial communication program after that.

This Bluetooth Serial is not suitable for sending large amounts of data.

[課題26] Make the RC servo to be controlled from PC via Bluetooth.(Finally, it became Radio Controlled!)

10.2 Wi-Fi (ESP32 configured as access point)

Many of the Wi-Fi samples requires external Wi-Fi router. However, ESP32 can be configured as a Wi-Fi access point (but it is slow). This method is used first because it can be used without a router.

Figure. 30and Figure. 31 shows the sample programs of ESP32 and Processing. In the ESP32 program, an infinite loop is created not by the loop function but by the while statement in it.

http://mukujii.sakura.ne.jp/esp2.html

```
#include <WiFi.h>

static const char *ssid = "ESP32Wifi"; //Change here
static const char *passwd = "password"; //Change here
const IPAddress ip(192, 168, 0, 33); //Change here
const IPAddress netmask(255, 255, 255, 0);
WiFiServer server(5204);  //Port id. Change here

void setup() {
  //Serial for error tracking.
  Serial.begin(115200);
  //Start WiFi access point
  WiFi.mode(WIFI_AP);
  WiFi.softAP(ssid, passwd);
  delay(100); //wait for event SYSTEM_EVENT_AP_START
  WiFi.softAPConfig(ip, ip, netmask);
  IPAddress myIP = WiFi.softAPIP();
  Serial.print("AP Started. myIP address: ");
  Serial.println(myIP);
  server.begin();
  Serial.println("Server started");
}
```

```
void loop() {
  char rcv;

  WiFiClient client = server.available();
  if(client){
    Serial.printf("New Client");
    while (client.connected()) { //Main loop is here.
      if(client.available()){
        rcv = client.read();
        client.write(rcv+10);
      }
    }
    client.stop();
    Serial.println("Client Disconnected.");
  }
  delay(100);
}
```

Figure. 30　Wi-Fi sample program (ESP32 side)

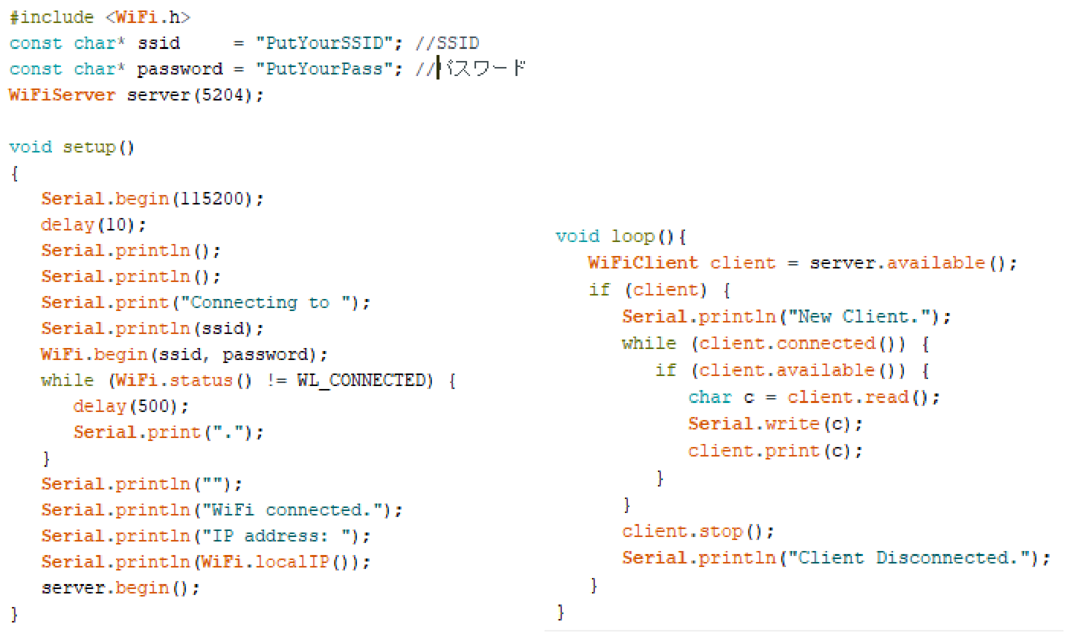
```
import processing.net.*;

Client myClient;

void setup() {
  print("Welcome to Wifi Access Point sample code.¥n");
  myClient = new Client(this, "192.168.0.33", 5204);
}

int c=0;

void draw() {
  int t;

  c=(c+1)%100;

  myClient.write(c);
  t = myClient.read();
  println("sent:"+c+"received:"+t);

}
```

Figure. 31　Wi-Fi sample program (Processing side)

[課題27]　Try to run the above sample program. change IP address, access point name, password, etc. arbitrarily. after writing ESP32, find Wi-Fi access point from PC and connect to it. After that, try to run Processing. Report the result of the operation and its discussion, and monitor it by serial communication on PC side. Also try to make ESP32 completely wireless by using mobile battery.

10.3 Wi-Fi (using an external router)

The method in the previous section is simple, but the PC must access the ESP32 access point, and the Internet connection cannot be used otherwise. Next, we try the method using an external Wi-Fi router.



```
#include <WiFi.h>
const char* ssid     = "PutYourSSID"; //SSID
const char* password = "PutYourPass"; //パスワード
WiFiServer server(5204);

void setup()
{
   Serial.begin(115200);
   delay(10);
   Serial.println();
   Serial.println();
   Serial.print("Connecting to ");
   Serial.println(ssid);
   WiFi.begin(ssid, password);
   while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
   }
   Serial.println("");
   Serial.println("WiFi connected.");
   Serial.println("IP address: ");
   Serial.println(WiFi.localIP());
   server.begin();
}
```

```
void loop(){
   WiFiClient client = server.available();
   if (client) {
      Serial.println("New Client.");
      while (client.connected()) {
         if (client.available()) {
            char c = client.read();
            Serial.write(c);
            client.print(c);
         }
      }
      client.stop();
      Serial.println("Client Disconnected.");
   }
}
```

Figure 32 Wi-Fi sample program Part 2 (ESP32 side)

[課題28] Try to run the above sample program. change the IP address, the name of the access point, the password and so on arbitrarily. It is necessary to check the number with serial terminal first. You may need to change the port number depending on your router configuration.

11 References

Here are some references other than those discussed in the text.

Links that were particularly helpful in the preparation of the document

[1] ESP-WROOM-32 https://ht-deko.com/arduino/esp-wroom-32.html

[2] ESP-IDF environment, unlike Arduino environment, you can use all features of ESP32. https://www.mgo-tec.com/esp32-idf-howto-01

book

[1] A standard textbook for "A Reintroduction to Operational Amplifier Basic Circuits".

[2] Ready to use! Operational Amplifier Schematics
[3] Transistor Technology SPECIAL "Practical Circuit Design with OP Amplifier
[4] Transistor Technology SPECIAL "OP Amp IC Application Note
[5] Transistor Technology SPECIAL "Thorough Illustration Digital Oscilloscope Application Note" SPECIAL series from CQ Publishing. The contents are standard, and it is for the next step for those who have graduated from the introductory level, and it is the type of book that can be kept in your desk for a long time. Other than this, all the Transistor Technology SPECIAL books in my laboratory are good reference books.
[6] Transistor Technology We subscribe monthly in the lab.