



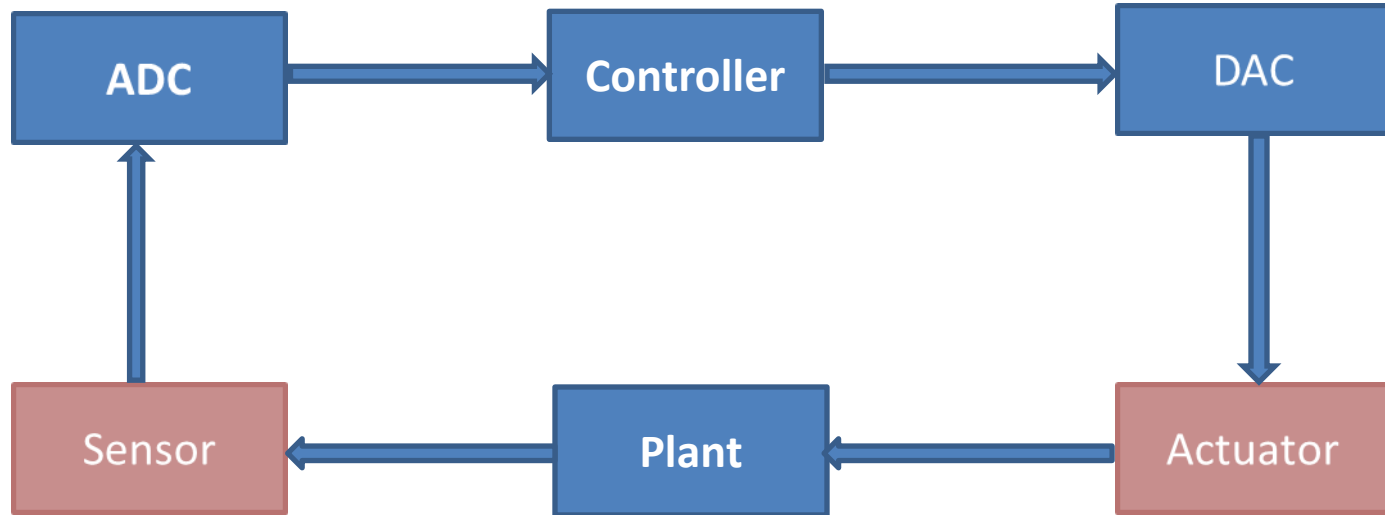
Introduction to Embedded Systems, Lectures 7, 8

Peripheral devices of ES: sensors and activators

Dmitry Zaitsev

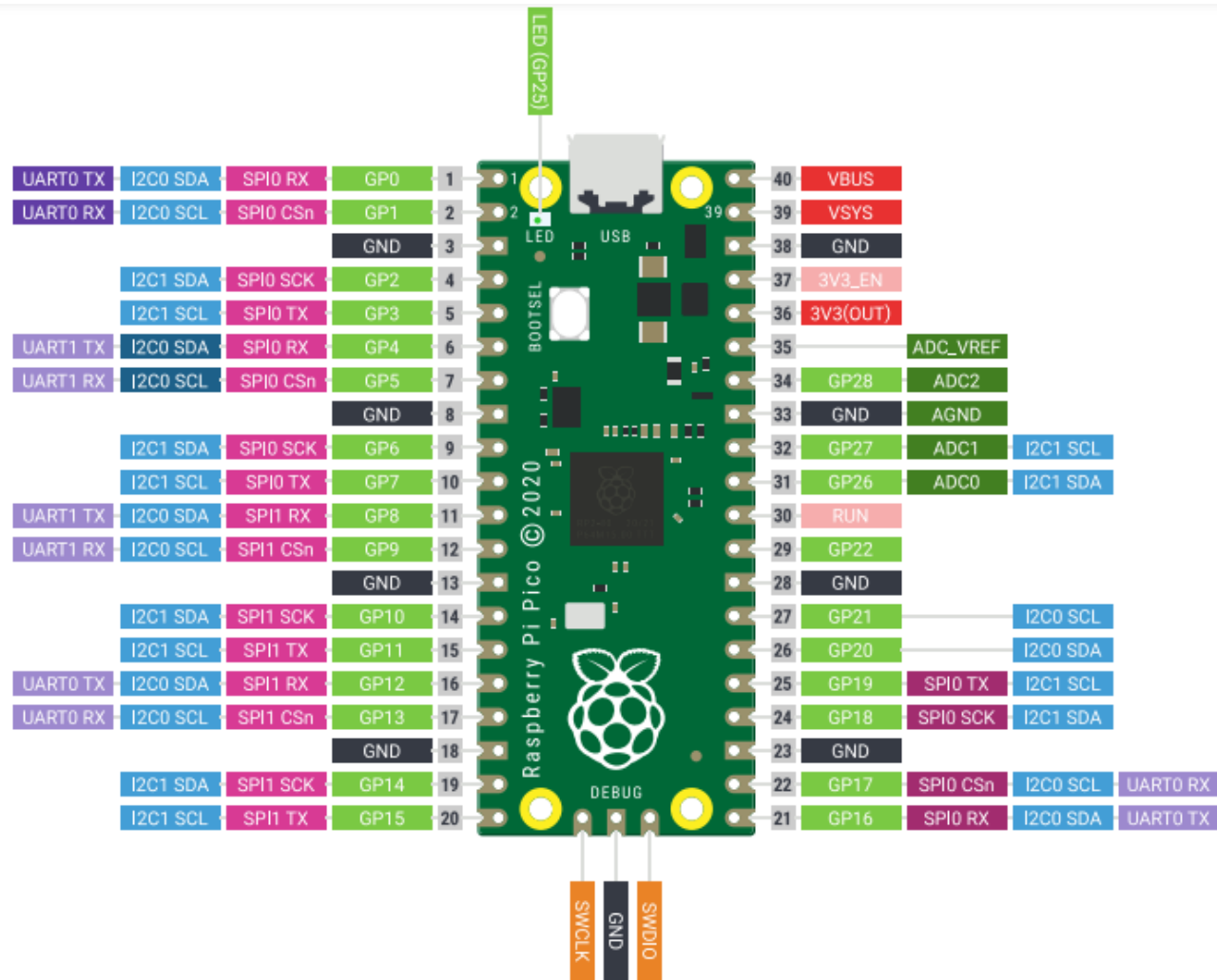
<http://daze.ho.ua>

General Scheme of ES



RP Pico pins

■	Power
■	Ground
■	UART / UART (default)
■	GPIO, PIO, and PWM
■	ADC
■	SPI / SPI (default)
■	I2C / I2C (default)
■	System Control
■	Debugging



RP Pico Pins

- **Ground pins: 8, rectangular, both sides**
- **Power pins: VBUS, VSYS, 3V3, 3V3_EN, RUN, right side**
- **GPIO pins GP0-GP28: 26 exposed, GP22-GP26 internal, for instance GP25 for LED**
- **Analog pins: 12-bit ADC0-ADC2 (GP26-GP28), internal ADC for temperature; for external use ADC_VREF, ADC_GND**

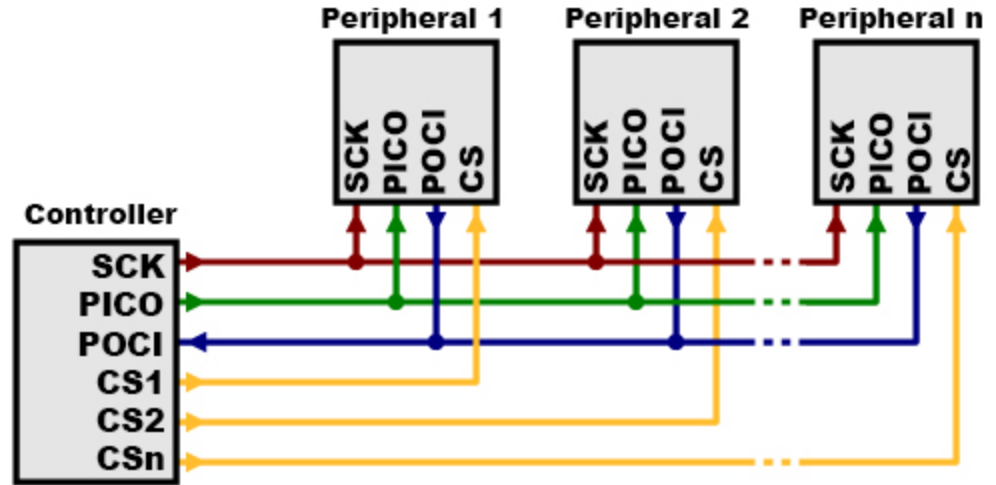
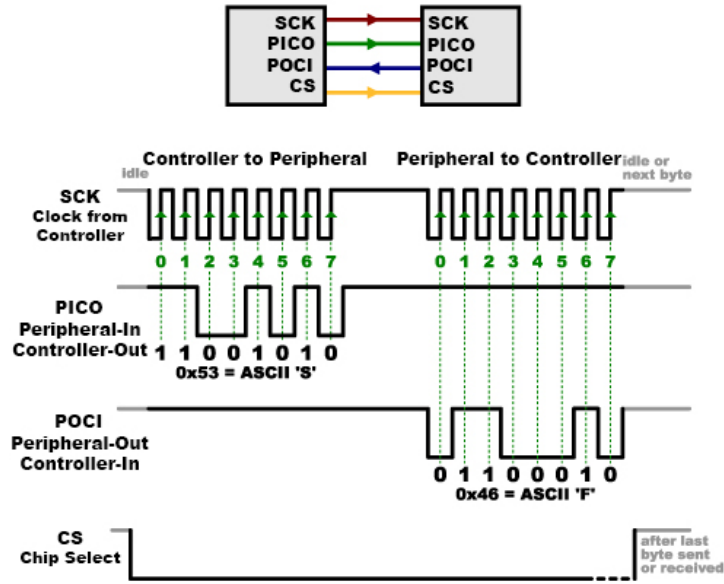
Power Pins

- **VBUS** – power from the microUSB bus, 5-volts. If the Pico is not being powered by the microUSB connector then there will be no output here.
- **VSYS** – the input voltage, which can range from 2 to 5-volts. The on-board voltage converter will change it to 3.3-volts for the Pico.
- **3V3** – a 3.3-volt output, from the Pico's internal regulator. It can be used to power additional components, providing you keep the load under 300ma.
- **3V3_EN** – disables the Pico's internal voltage regulator, which will shut off the Pico and any components powered by it.
- **RUN** – enables or disables the RP2040 microcontroller, or resets it.

GPIO Pins

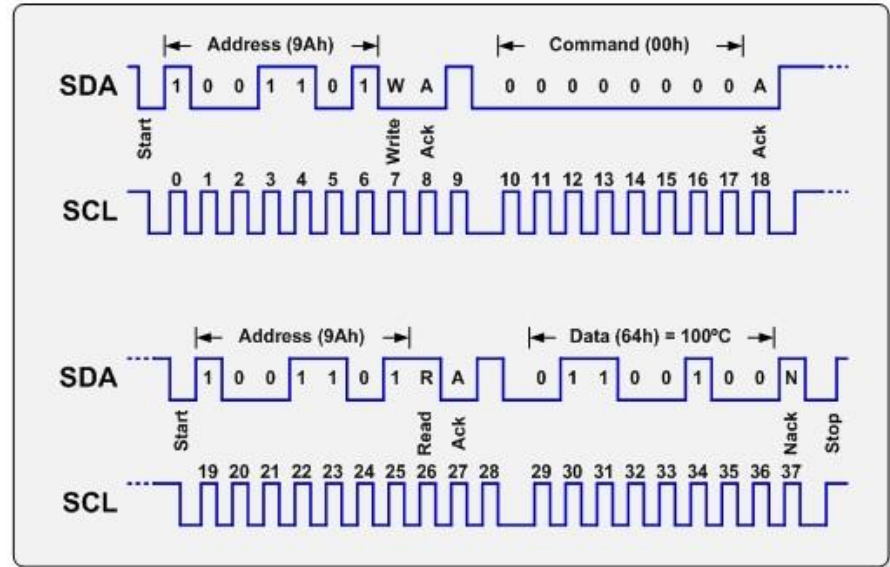
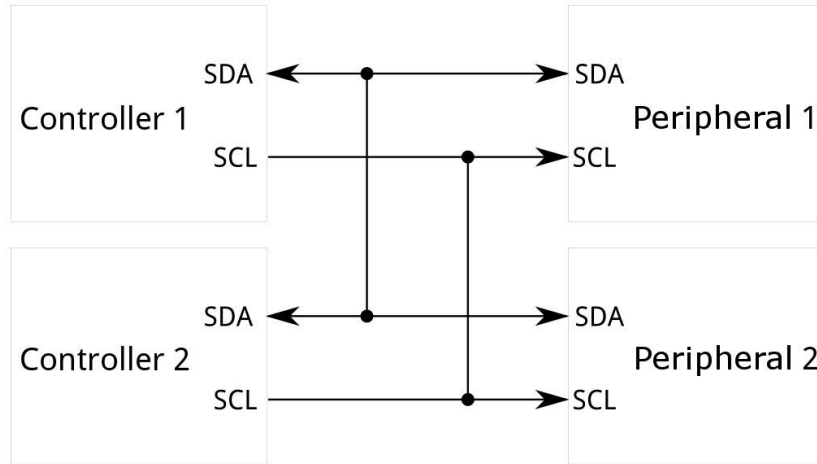
- **2 × SPI, Serial Peripheral Interface**
- **2 × I2C, Inter-Integrated Circuit**
- **2 × UART, Universal Asynchronous Receiver-Transmitter**
- **16 × controllable PWM channels, Pulse Width Modulation**

Multi-peripheral access with SPI

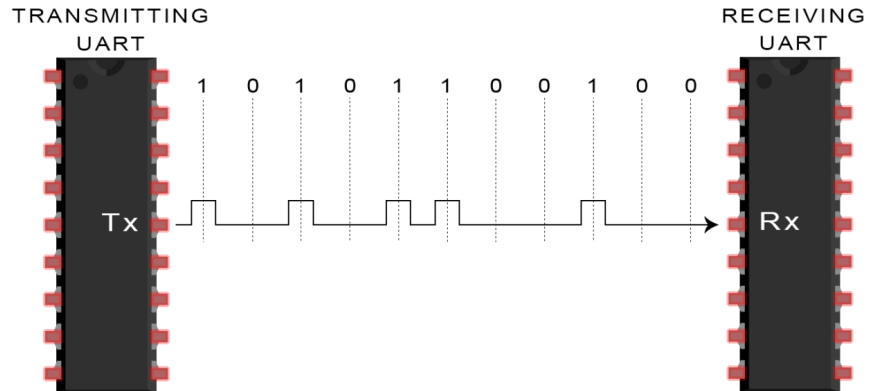
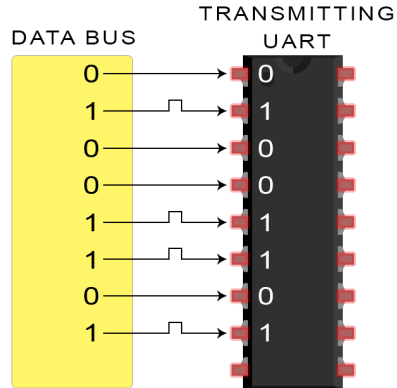
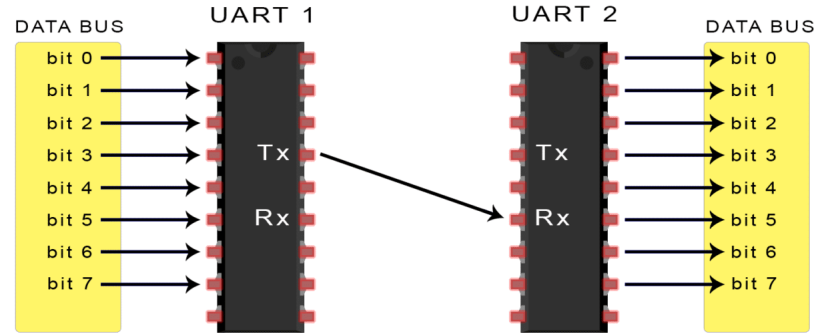
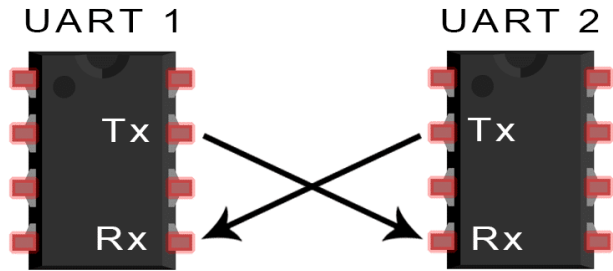


Inter-Integrated Circuit

- for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communication

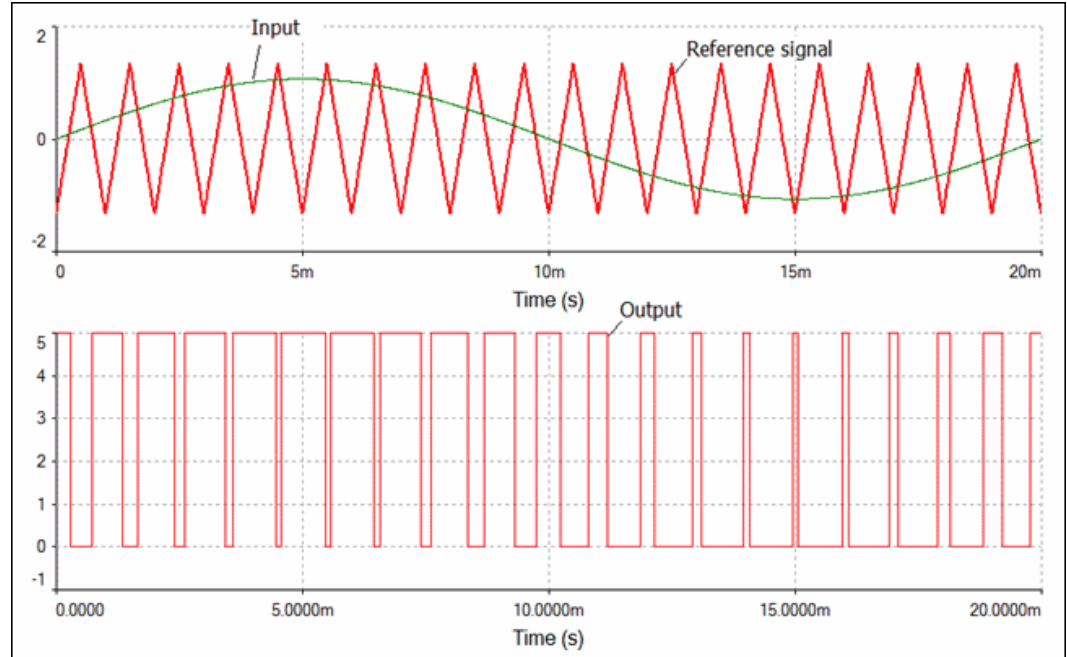


Universal Asynchronous Receiver-Transmitter



Pulse Width Modulation

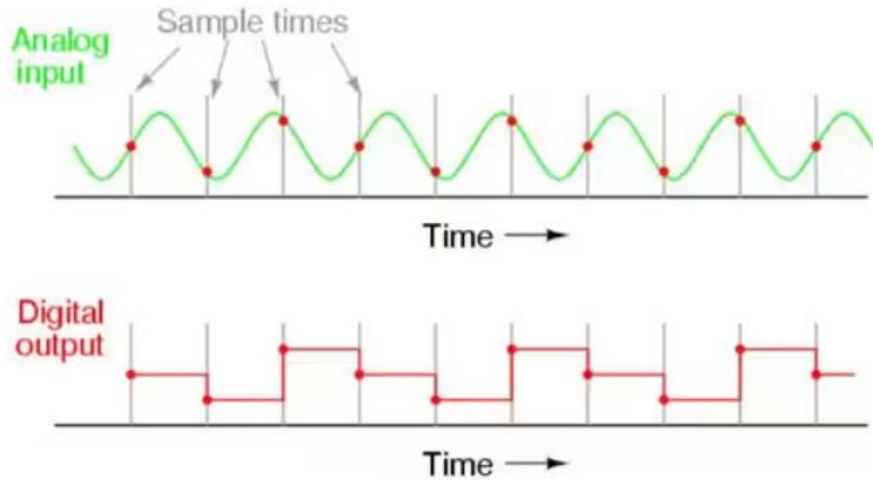
- Pulse width modulation (PWM) is a modulation technique that generates variable-width pulses to represent the amplitude of an analog input signal.



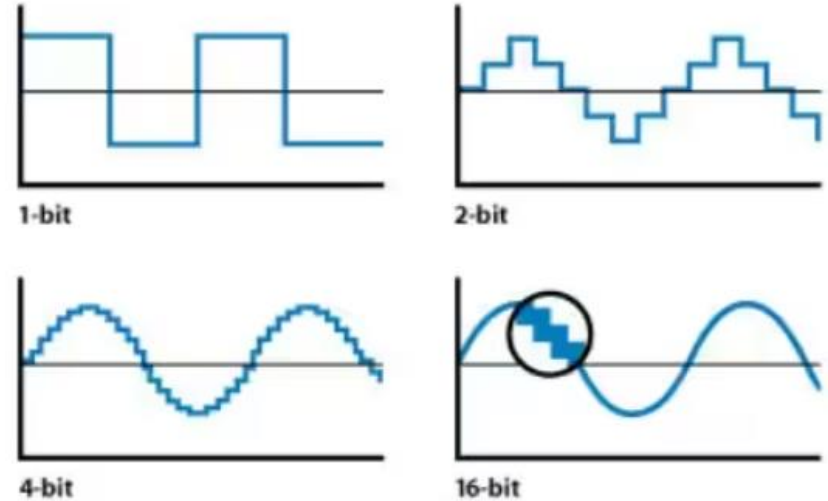
Analog Pins

- Analog to Digital Converter (ADC) converts an analog voltage on a pin to a digital number.
- 12 Bits ADC with a quantization level of 4096
- A/D conversion in polling, interrupt & FIFO with DMA mode
- The ADC conversion speed per sample is $2\mu\text{s}$ that is 500kS/s
- ADC converters usually generate interrupts when a conversion is completed

Analog to Digital Conversion



Sequence of values






























































Resolution

Actuators and Sensors of RP Pico Starter Kit

Download Needed:
Tutorials and Code
(No paper tutorial.)

The download link can be found on the box!

<div>Download Needed: Tutorials and Code (No paper tutorial.)</div>		<div>Official Raspberry Pi Pico</div> <div>x1</div>		<div></div> <div>micro:bit GPIO Extension Board</div> <div>x1</div>		<div></div> <div>I2C LCD 1602</div> <div>x1</div>		<div></div> <div>Resistor-220</div> <div>x20</div>									
<div>The download link can be found on the box!</div>		<div></div> <div>Potentiometer</div> <div>x3</div>		<div></div> <div>Capacitor 0.1uf</div> <div>x2</div>		<div></div> <div>Capacitor 10uf</div> <div>x2</div>		<div></div> <div>Push Button</div> <div>x4</div>		<div></div> <div>Big Push Button</div> <div>x4</div>		<div></div> <div>Red Push Button Cap</div> <div>x1</div>		<div></div> <div>Green Push Button Cap</div> <div>x1</div>		<div></div> <div>Resistor-1K</div> <div>x10</div>	
<div></div> <div>Switch</div> <div>x2</div>		<div></div> <div>Vibration Switch</div> <div>x1</div>		<div></div> <div>Keypad</div> <div>x1</div>		<div></div> <div>1N4001 Diode</div> <div>x2</div>		<div></div> <div>1N4148 Diode</div> <div>x2</div>		<div></div> <div>Blue Push Button Cap</div> <div>x1</div>		<div></div> <div>Yellow Push Button Cap</div> <div>x1</div>		<div></div> <div>Red Led</div> <div>x10</div>			
<div></div> <div>Photoresistor</div> <div>x1</div>		<div></div> <div>4-Digit 7-Segment Display</div> <div>x1</div>		<div></div> <div>Passive Buzzer</div> <div>x1</div>		<div></div> <div>Active Buzzer</div> <div>x1</div>		<div></div> <div>L293D</div> <div>x1</div>		<div></div> <div>74HC595</div> <div>x2</div>		<div></div> <div>8 RGB LED Module</div> <div>x1</div>		<div></div> <div>Green Led</div> <div>x4</div>			
<div></div> <div>DHT-11</div> <div>x1</div>		<div></div> <div>Thermistor</div> <div>x1</div>		<div></div> <div>7-Segment Display</div> <div>x1</div>		<div></div> <div>Motor</div> <div>x1</div>		<div></div> <div>Servo</div> <div>x1</div>		<div></div> <div>Stepping Motor</div> <div>x1</div>		<div></div> <div>RGB Led</div> <div>x1</div>		<div></div> <div>Blue Led</div> <div>x4</div>			
<div></div> <div>Joystick</div> <div>x1</div>		<div></div> <div>Infrared Motion Sensor</div> <div>x1</div>		<div></div> <div>Ultrasonic Ranging Module</div> <div>x1</div>		<div></div> <div>Stepping Motor Driver</div> <div>x1</div>		<div></div> <div>9V Battery Cable</div> <div>x1</div>		<div></div> <div>65 Jump Wire M-M</div> <div>x1</div>		<div></div> <div>Breadboard</div> <div>x1</div>		<div></div> <div>Yellow Led</div> <div>x4</div>			
<div></div> <div>10 Jump Wire F-F</div> <div>x1</div>		<div></div> <div>10 Jump Wire F-M</div> <div>x1</div>		<div></div> <div>NPN Transistor 8050</div> <div>x2</div>		<div></div> <div>PNP Transistor 8550</div> <div>x2</div>		<div></div> <div>Infrared Remote</div> <div>x1</div>		<div></div> <div>8*8 LEDMatrix</div> <div>x1</div>		<div></div> <div>RFID Module</div> <div>x1</div>		<div></div> <div>Led Bar Graph</div> <div>x1</div>			
<div></div> <div>Relay</div> <div>x1</div>		<div></div> <div>BreadBoardPower</div> <div>x1</div>		<div></div> <div>MPU6050</div> <div>x1</div>		<div></div> <div>General Board</div> <div>x3</div>		<div></div> <div>Pinout Card</div> <div>x1</div>		<div></div> <div>Pinout Sticker</div> <div>x1</div>		<div></div> <div>Resistor Color Code Card</div> <div>x1</div>		<div></div> <div>Plastic Box</div> <div>x1</div>			

Sensors of RP Pico Starter Kit

- photoresistor
- thermistor
- potentiometer
- button
- joystick
- serial input
- ultrasonic ranging
- matrix keypad
- infrared remote control
- infrared motion detector
- attitude sensor
- RFID

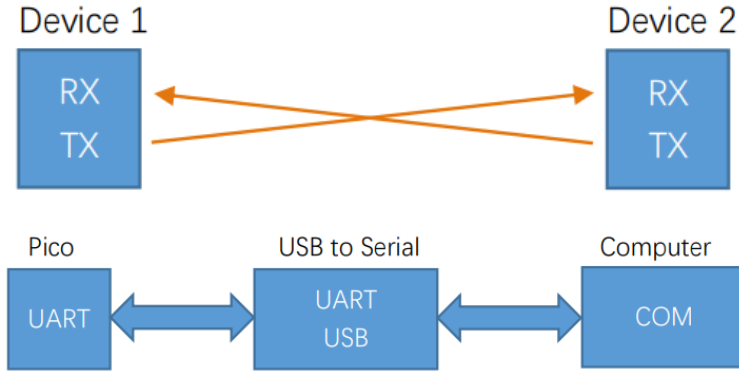
Actuators of RP Pico Starter Kit

- relay & motor
- servo
- stepper motor
- buzzer
- segment display
- LCD Display Screen
- 74HC595 IC Chip
- serial write
- lamps:
 - LED
 - LED bar
 - RGBLED
 - 8 RGB LED

Basic Arduino IDE library routines

- `pinMode(PIN, INPUT/OUTPUT/ INPUT_PULLUP);`
- `val = digitalRead(PIN);`
- `digitalWrite(PIN, LOW/HIGH);`
- `val = analogRead(PIN_ADC);`
- `analogWrite(PIN,val);`

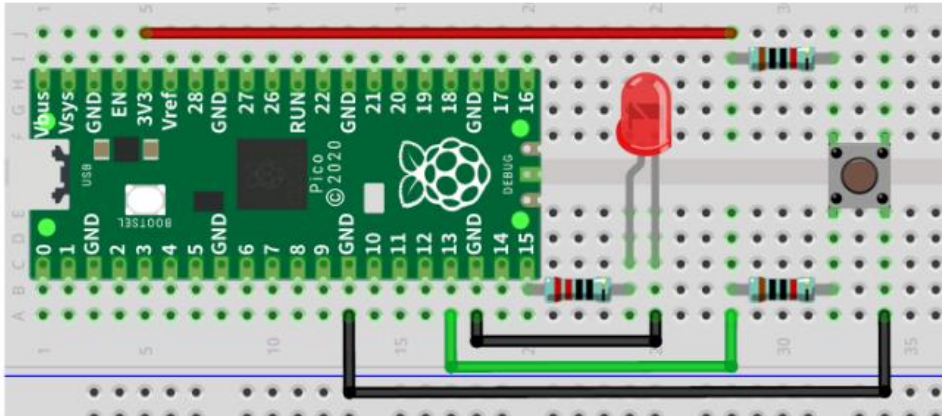
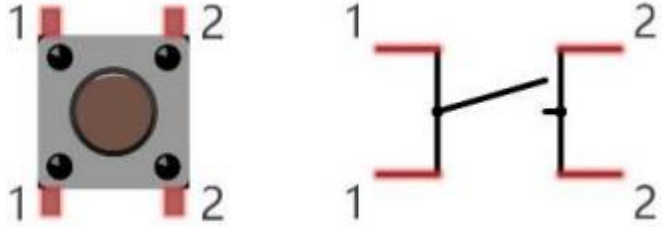
Serial input/output



```
String inputString = "";  
bool stringComplete = false;  
void setup() {  
  Serial.begin(115200);  
  delay(1000);  
  Serial.println(String("Input data: "));  
}
```

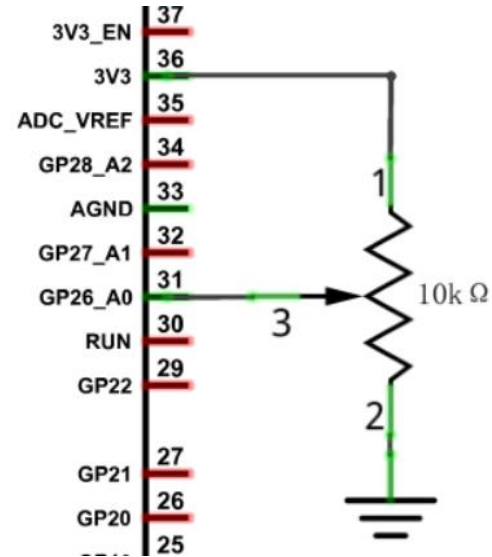
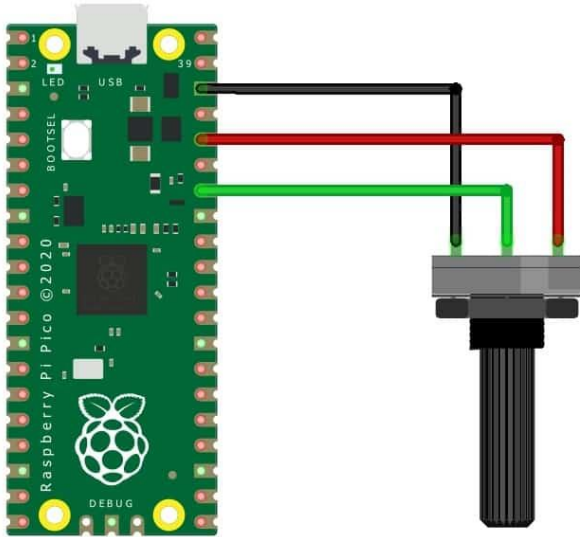
```
void loop() {  
  if (Serial.available()) {  
    char inChar = Serial.read();  
    inputString += inChar;  
    if (inChar == '\n') {  
      stringComplete = true;  
    }  
    if (stringComplete) {  
      Serial.print("input text: ");  
      Serial.print(inputString);  
      inputString = "";  
      stringComplete = false;  
    }  
  }  
}
```

Button



- Initialization:
`pinMode(PIN_BUTTON, INPUT);`
- Reading:
`digitalRead(PIN_BUTTON)`
- Range: LOW, HIGH

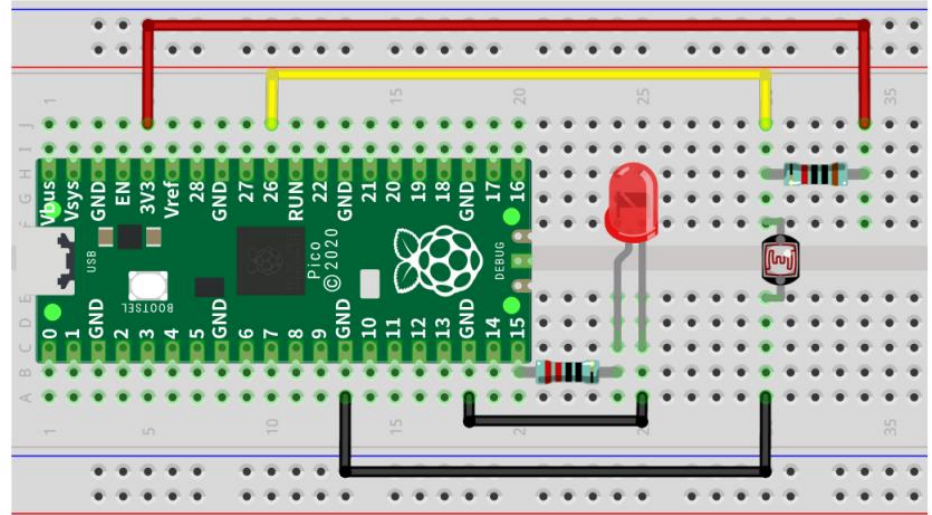
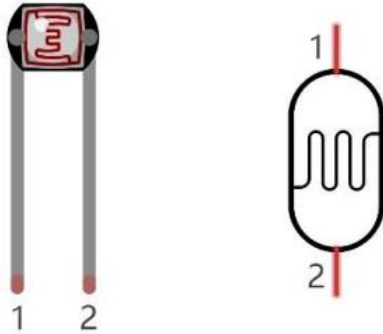
Potentiometer



- Read ADC:

```
int adcVal = analogRead(PIN_ADC0);
```

Photoresistor



- Read ADC:

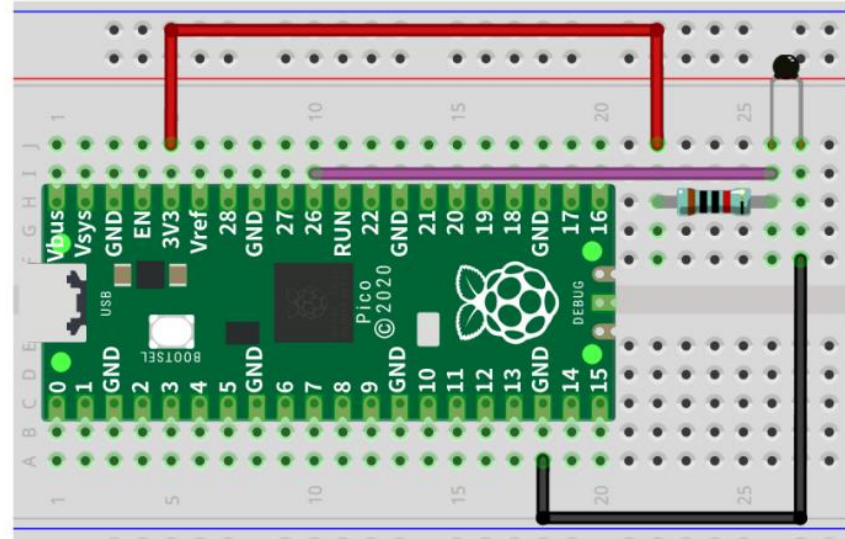
```
int adcVal = analogRead(PIN_ADC0);
```



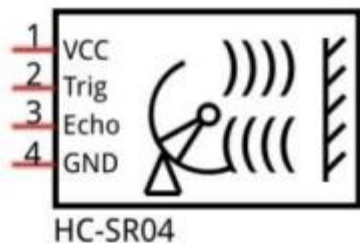
Thermistor

$$T2 = 1 / \left(\frac{1}{T1} + \ln\left(\frac{Rt}{R}\right) / B \right)$$

- Read ADC:
`int adcValue = analogRead(PIN_ADC0);`



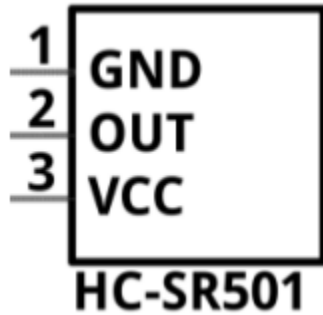
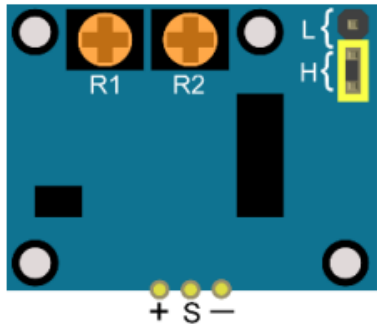
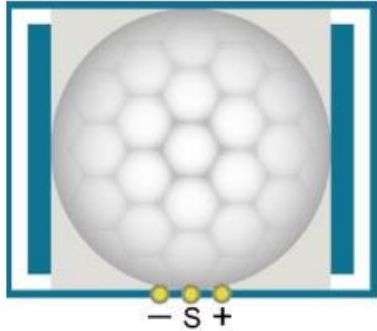
Ultrasonic ranging



Pin	Description
VCC	power supply pin
Trig	trigger pin
Echo	Echo pin
GND	GND

```
float getSonar()
{
    unsigned long pingTime;
    float distance;
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    pingTime = pulseIn(echoPin, HIGH, timeOut);
    distance = (float)pingTime * soundVelocity / 2 /
                10000;
    return distance;
}
```

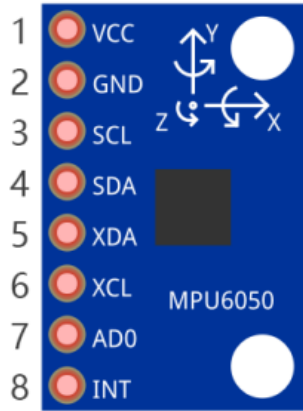
Infrared motion detector



```
void setup(){  
  pinMode(IMD_PIN,INPUT);  
  pinMode(LED_PIN,OUTPUT);  
}
```

```
void loop(){  
  v=digitalRead(IMD_PIN);  
  digitalWrite(LED_PIN,v);  
  delay(1000);  
}
```

Attitude sensor



```
mpu6050.begin(); //initialize the MPU6050
```

```
mpu6050.calcGyroOffsets(true); //get the offsets value
```

```
ax=mpu6050.getRawAccX(); //gain the values of X axis acceleration raw data
```

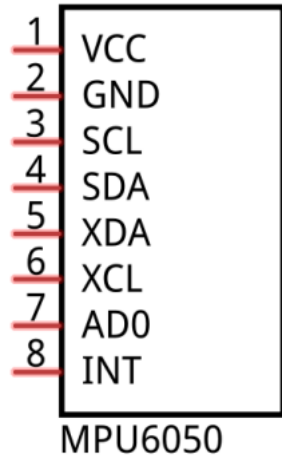
```
ay=mpu6050.getRawAccY(); //gain the values of Y axis acceleration raw data
```

```
az=mpu6050.getRawAccZ(); //gain the values of Z axis acceleration raw data
```

```
gx=mpu6050.getRawGyroX(); //gain the values of X axis Gyroscope raw data
```

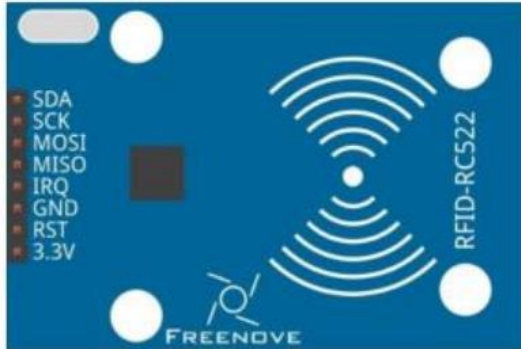
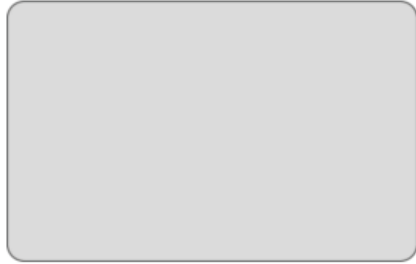
```
gy=mpu6050.getRawGyroY(); //gain the values of Y axis Gyroscope raw data
```

```
gz=mpu6050.getRawGyroZ(); //gain the values of Z axis Gyroscope raw data
```



The port description of the MPU6050 module is as follows:

Pin name	Pin number	Description
VCC	1	Positive pole of power supply with voltage 5V
GND	2	Negative pole of power supply
SCL	3	I2C communication clock pin
SDA	4	I2C communication clock pin
XDA	5	I2C host data pin which can be connected to other devices.
XCL	6	I2C host clock pin which can be connected to other devices.
AD0	7	I2C address bit control pin. Low level: the device address is 0x68 High level: the device address is 0x69
INT	8	Output interrupt pin

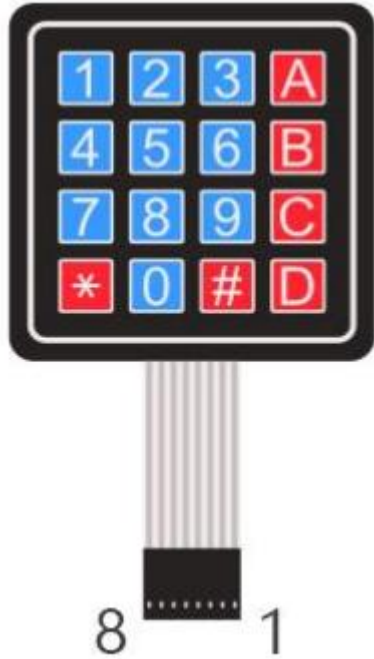


RFID

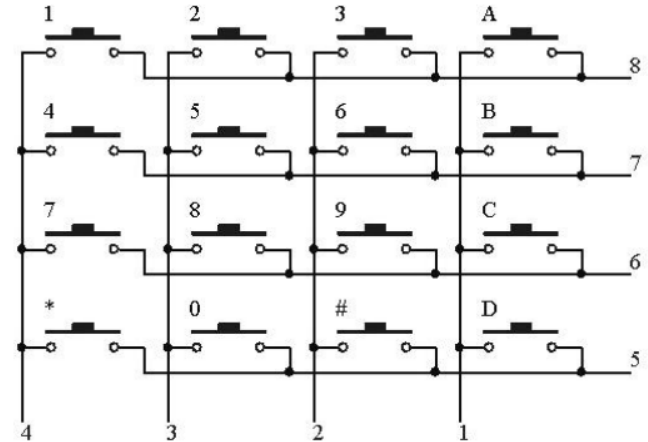
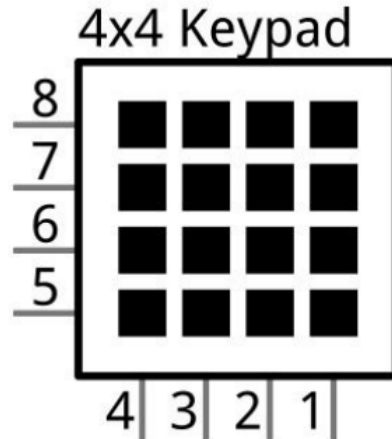
```
SPI.begin();  
rfid.init();  
rfid.findCard(PICC_REQIDL, str)  
rfid.anticoll(str)
```

Sector No.	Block No.	Storage area	Block type	Absolute block No.
sector 0	block 0	vendor code	vendor block	0
	block 1		data block	1
	block 2		data block	2
	block 3	Password A-access control-password B	control block	3
sector 1	block 0		data block	4
	block 1		data block	5
	block 2		data block	6
	block 3	Password A-access control-password B	control block	7
.....	
sector 15	block 0		data block	60
	block 1		data block	61
	block 2		data block	62
	block 3	Password A-access control-password B	control block	63

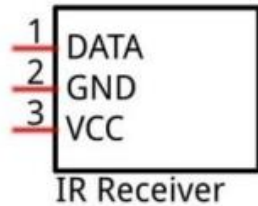
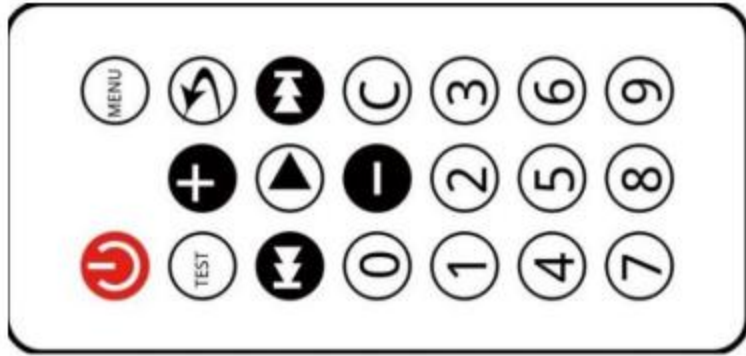
Matrix keypad



```
pinMode(pin, OUTPUT);  
digitalWrite(pin, LOW/HIGH);  
pinMode(pin, INPUT_PULLUP);  
digitalRead(pin)
```



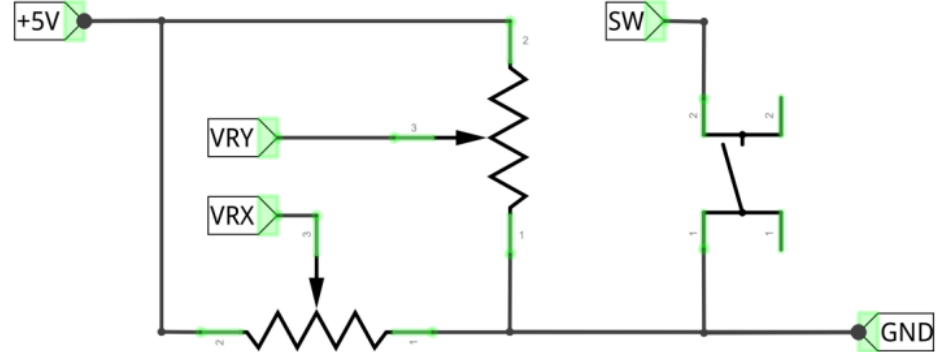
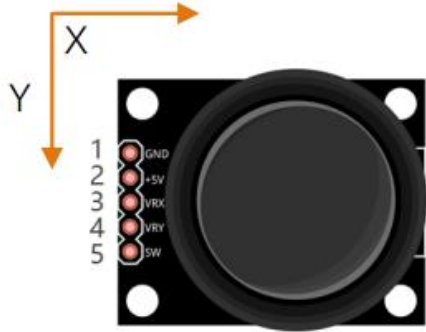
Infrared remote control



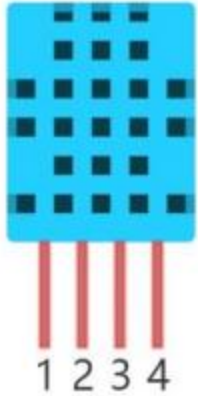
ICON	KEY Value	ICON	KEY Value
	FFA25D		FFB04F
	FFE21D		FF30CF
	FF22DD		FF18E7
	FF02FD		FF7A85
	FFC23D		FF10EF
	FFE01F		FF38C7
	FFA857		FF5AA5
	FF906F		FF42BD
	FF6897		FF4AB5
	FF9867		FF52AD

Joystick

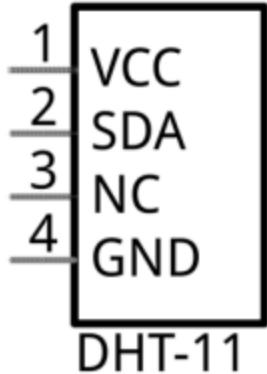
```
pinMode(xyzPins[2], INPUT_PULLUP);  
int xVal = analogRead(xyzPins[0]);  
int yVal = analogRead(xyzPins[1]);  
int zVal = digitalRead(xyzPins[2]);
```



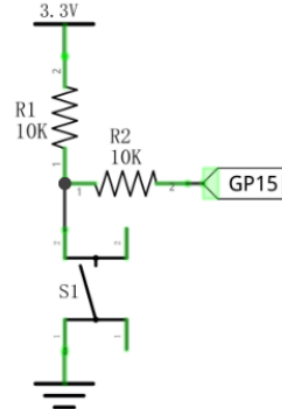
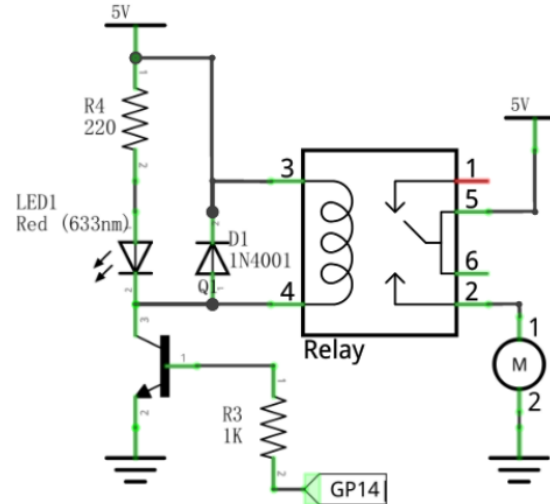
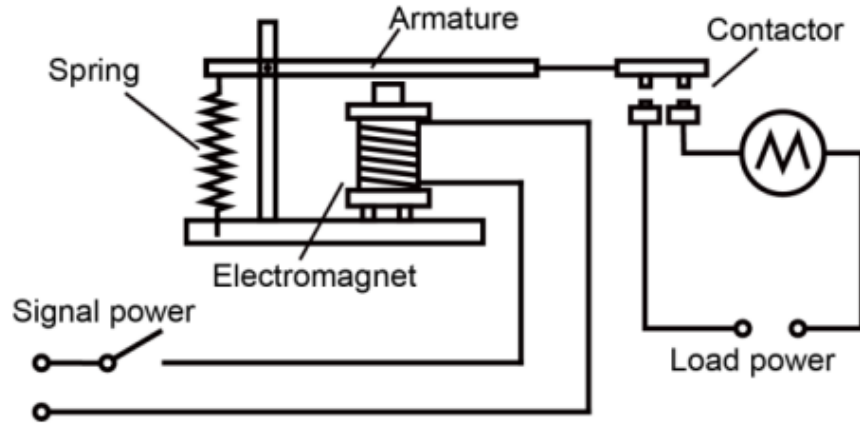
Hygrothermograph



```
void loop(){
  int chk = DHT.read11(DHT11_PIN);
  if(chk == DHTLIB_OK){
    Serial.println("humidity: " + String(DHT.humidity) +
      "%, temperature: " +
      String(DHT.temperature) + "C");
  }else{
    Serial.println("DHT11 Reading data error!");
  }
  delay(2000);
}
```



Relay and Motor (1)



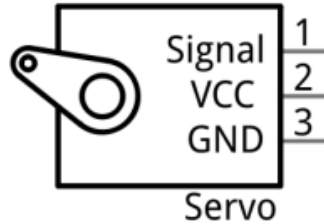
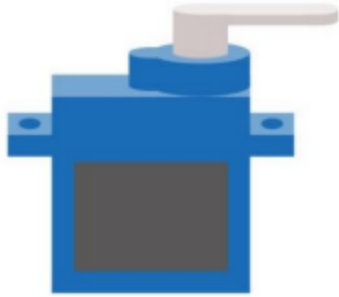
Relay and Motor (2)

```
int buttonState = HIGH;
int relayState = LOW;
int lastButtonState = HIGH;
long lastChangeTime = 0;

void setup() {
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, relayState);
}
```

```
void loop() {
  int nowButtonState = digitalRead(buttonPin);
  if (nowButtonState != lastButtonState)
    lastChangeTime = millis();
  if (millis() - lastChangeTime > 10) {
    if (buttonState != nowButtonState) {
      buttonState = nowButtonState;
      if (buttonState == LOW) {
        relayState = ! relayState;
        digitalWrite(relayPin, relayState); } } }
  lastButtonState = nowButtonState;
}
```

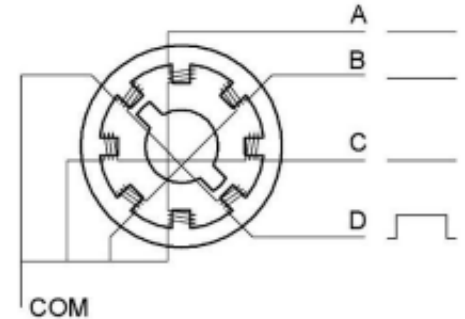
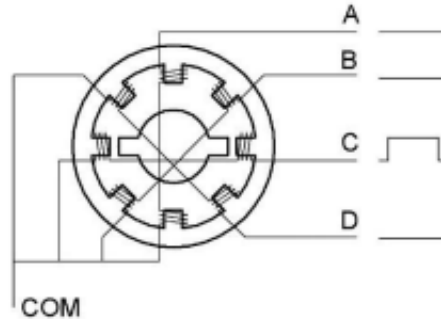
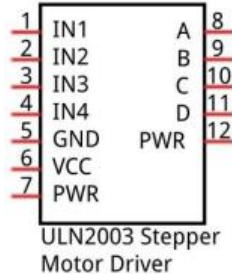
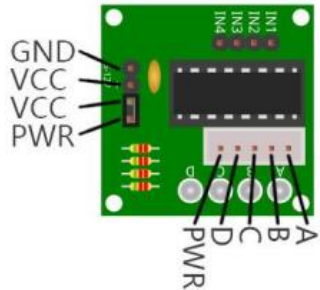
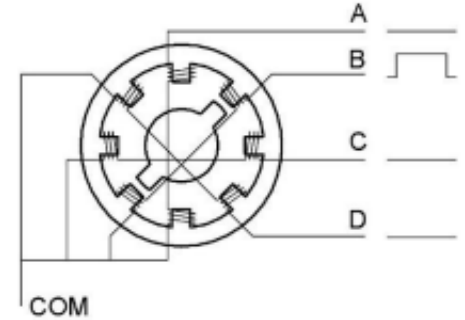
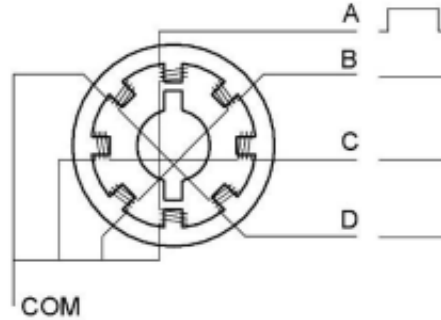
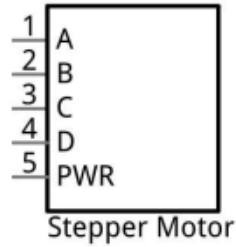
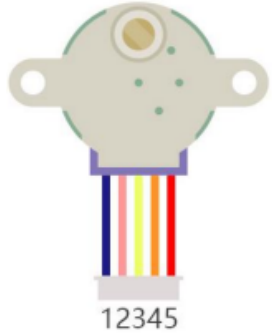
Servo Sweep



High level time	Servo angle
0.5ms	0 degree
1ms	45 degree
1.5ms	0 degree
2ms	45 degree
2.5ms	180 degree

```
#include <Servo.h>
#define servoPin 16
Servo myServo;
int pos = 0;
void setup() {
  myServo.attach(servoPin);
}
void loop() {
  for (pos = 0; pos <= 180; pos += 1) {
    myServo.write(pos);
    delay(15);
  }
  for (pos = 180; pos >= 0; pos -= 1) {
    myServo.write(pos);
    delay(15);
  }
}
```


Stepper Motor (1)

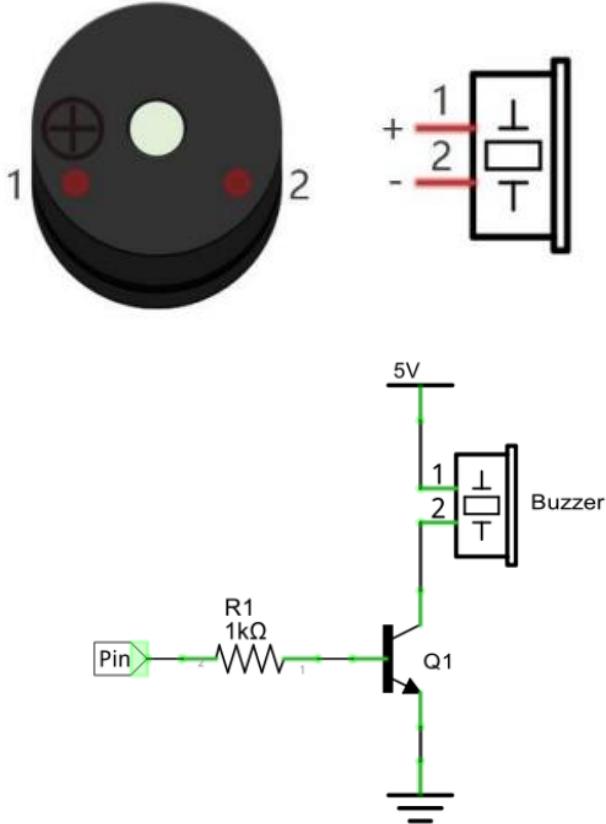


Stepper Motor (2)

```
int outPorts[] = {21, 20, 19, 18};  
void setup() {  
  for (int i = 0; i < 4; i++) {  
    pinMode(outPorts[i], OUTPUT);  
  }  
}  
void loop(){  
  moveSteps(true, 32 * 64, 3);  
  delay(1000);  
  moveSteps(false, 32 * 64, 3);  
  delay(1000);  
}
```

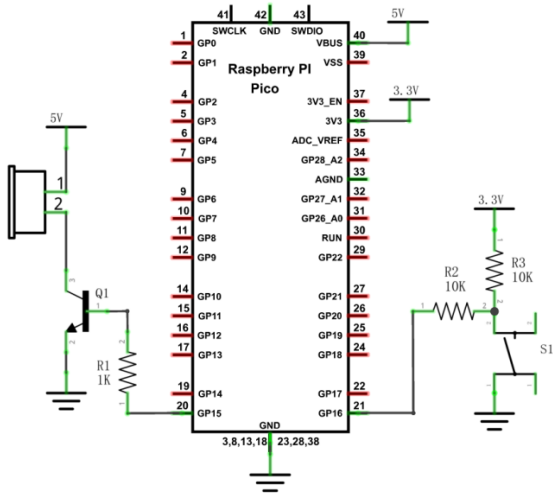
```
void moveSteps(bool dir, int steps, byte ms) {  
  for (unsigned long i = 0; i < steps; i++) {  
    moveOneStep(dir);  
    delay(constrain(ms,3,20)); }  
}  
void moveOneStep(bool dir) {  
  static byte out = 0x01;  
  if (dir) out != 0x08 ? out = out << 1 : out = 0x01;  
  else out != 0x01 ? out = out >> 1 : out = 0x08;  
  for (int i = 0; i < 4; i++) {  
    digitalWrite(outPorts[i], (out & (0x01 << i)) ? HIGH :  
    LOW); }  
}
```

Active Buzzer



```
void setup() {  
  pinMode(PIN_BUZZER, OUTPUT);  
  pinMode(PIN_BUTTON, INPUT);  
  digitalWrite(PIN_BUZZER, LOW);  
}  
  
void loop() {  
  if (digitalRead(PIN_BUTTON) == LOW) {  
    digitalWrite(PIN_BUZZER, HIGH);  
  } else {  
    digitalWrite(PIN_BUZZER, LOW);  
  }  
}
```

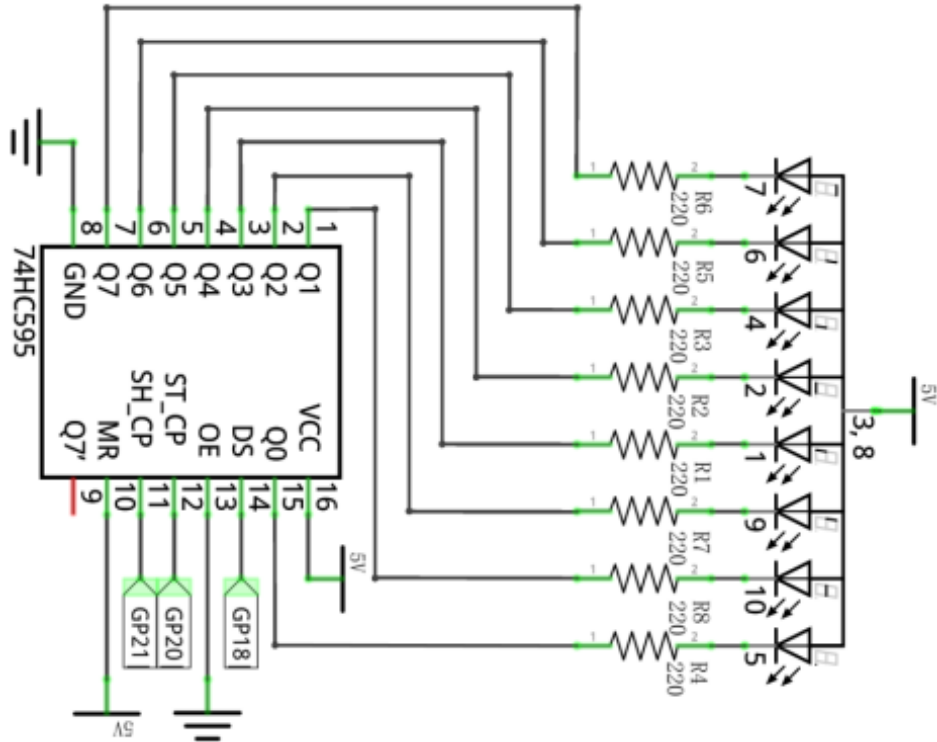
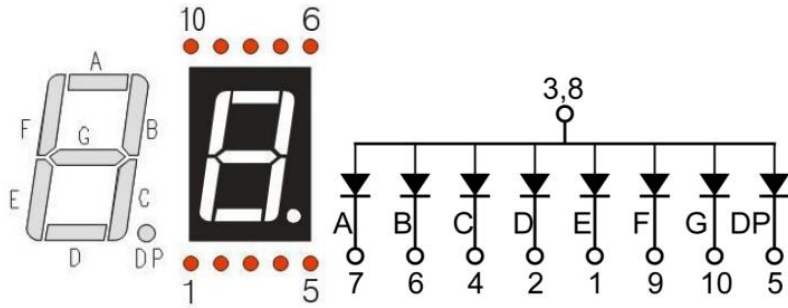
Passive Buzzer



```
void alert() {
    float sinVal;
    int toneVal;
    for (int x = 0; x < 360; x += 10) {
        sinVal = sin(x * (PI / 180));
        toneVal = 2000 + sinVal * 500;
        freq(PIN_BUZZER, toneVal, 10); }
}
```

```
void freq(int PIN, int freqs, int times) {
    if (freqs == 0)
        digitalWrite(PIN, LOW);
    else {
        for (int i = 0; i < times * freqs / 1000; i++) {
            digitalWrite(PIN, HIGH);
            delayMicroseconds(1000000 / freqs / 2);
            digitalWrite(PIN, LOW);
            delayMicroseconds(1000000 / freqs / 2);
        }
    }
}
```

Segment Display (1)



Segment Display (2)

```
int dataPin = 18;  
int latchPin = 20;  
int clockPin = 21;
```

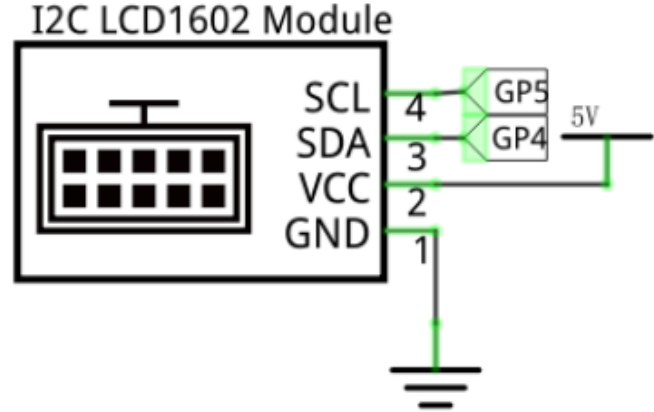
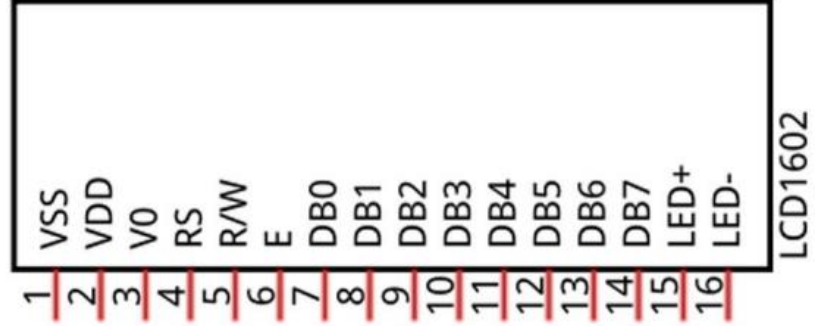
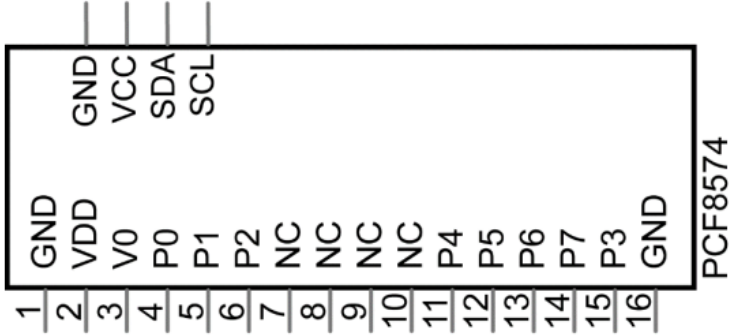
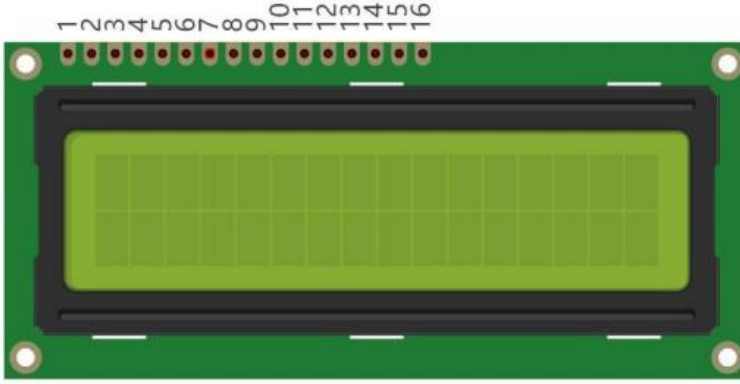
```
byte num[] = { 0xc0, 0xf9, 0xa4,  
0xb0, 0x99, 0x92, 0x82, 0xf8, 0x80,  
0x90, 0x88, 0x83, 0xc6, 0xa1, 0x86,  
0x8e };
```

```
void setup() {  
  pinMode(latchPin, OUTPUT);  
  pinMode(clockPin, OUTPUT);  
  pinMode(dataPin, OUTPUT);  
}
```

```
void loop() {  
  for (int i = 0; i < 16; i++) {  
    writeData(num[i]);  
    delay(1000);  
    wteData(0xff);  
  }  
}
```

```
void writeData(int value) {  
  digitalWrite(latchPin, LOW);  
  shiftOut(dataPin, clockPin, LSBFIRST, value);  
  digitalWrite(latchPin, HIGH);  
}
```

LCD Display Screen (1)



LCD Display Screen (2)

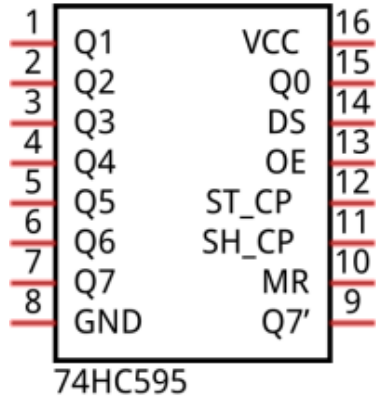
```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);

void setup() {
  if (!i2CAddrTest(0x27)) {
    lcd = LiquidCrystal_I2C(0x3F, 16, 2);
  }
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("hello world");
}
```

```
void loop() {
  lcd.setCursor(0,1);
  lcd.print("Counter:");
  lcd.print(millis() / 1000);
  delay(1000);
}

bool i2CAddrTest(uint8_t addr) {
  Wire.begin();
  Wire.beginTransmission(addr);
  if (Wire.endTransmission() == 0)
    return true; else return false;
}
```

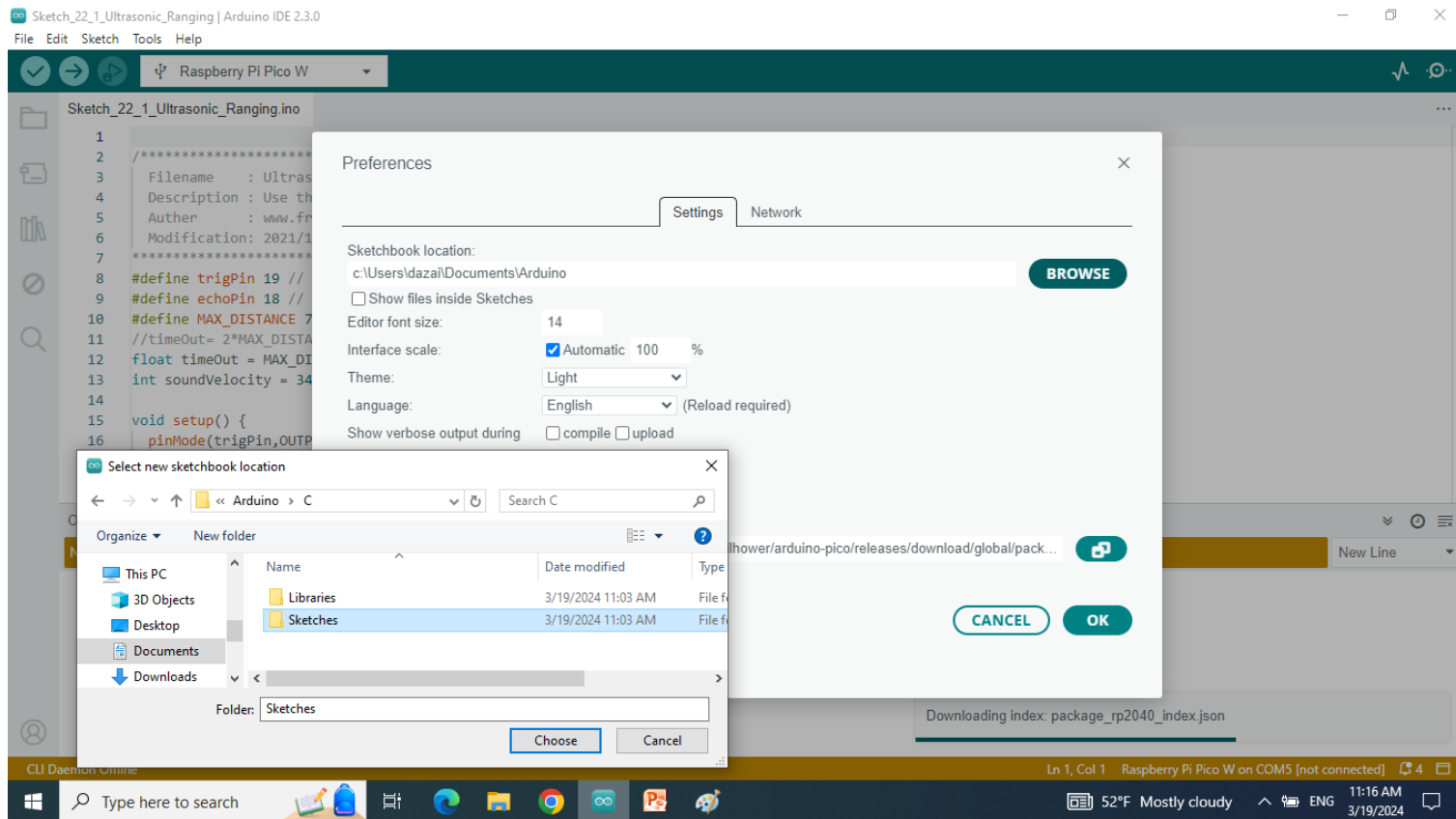

IC Chip 74HC595



- converts serial data into parallel data (octet)
- extends the number of RP Pico pins
- at least 3 pins are required to control 8 pins:
 - DS
 - ST_CP
 - SH_CP

Pin name	GPIO number	Description
Q0-Q7	15, 1-7	Parallel data output
VCC	16	The positive electrode of power supply, the voltage is 2~6V
GND	8	The negative electrode of power supply
DS	14	Serial data Input
OE	13	Enable output, When this pin is in high level, Q0-Q7 is in high resistance state When this pin is in low level, Q0-Q7 is in output mode
ST_CP	12	Parallel Update Output: when its electrical level is rising, it will update the parallel data output.
SH_CP	11	Serial shift clock: when its electrical level is rising, serial data input register will do a shift.
MR	10	Remove shift register: When this pin is in low level, the content in shift register will be cleared.
Q7'	9	Serial data output: it can be connected to more 74HC595 in series.

Adding sketchbooks in Arduino IDE



Using libraries in Arduino IDE

