Advanced System Software
（先端システムソフトウェア）

#3 (2018/10/11)

CSC.T431, 2018-3Q
Mon/Thu 9:00-10:30, W832

Instructor: Takuo Watanabe（渡部卓雄）
Department of Computer Science

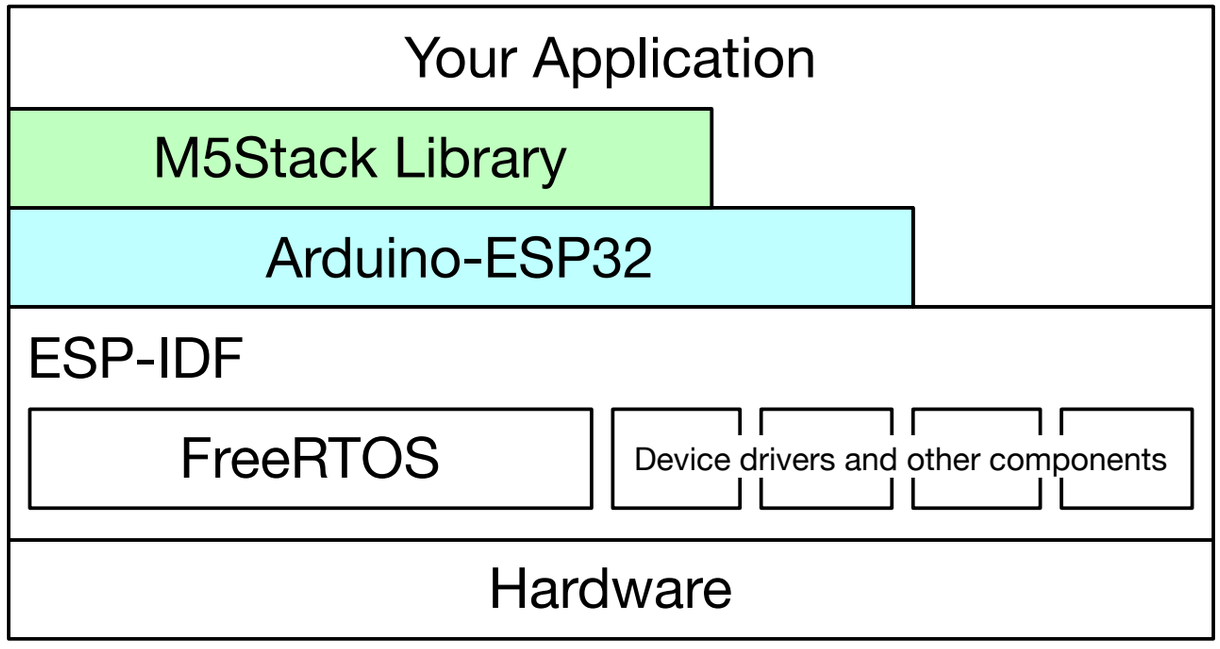
e-mail: takuo∅c.titech.ac.jp
http://www.psg.c.titech.ac.jp/~takuo/
ext: 3690, office: W8E-805

Agenda

- Programming Embedded Systems (1)
  - Programming M5Stack as Arduino

Programming M5Stack as Arduino

- Arduino core for ESP32 (Arduino-ESP32)
- M5Stack Library

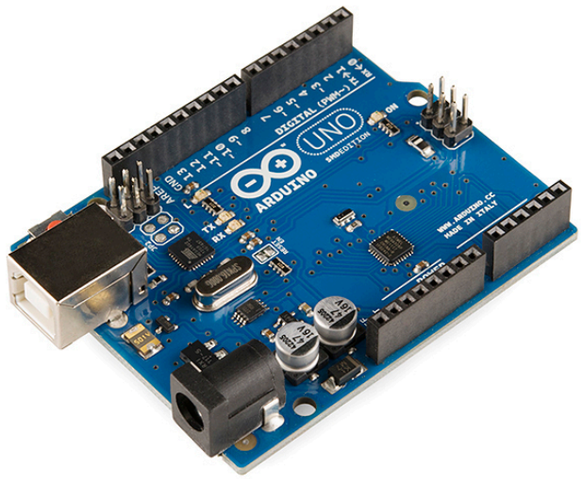


Arduino-ESP32

- Arduino library for ESP32-based devices
  - https://github.com/espressif/arduino-esp32
  - General Arduino API
    - https://www.arduino.cc
    - https://www.arduino.cc/reference/en/
  - ESP32-specific library components

Arduino

https://www.arduino.cc



- An open-source microcontroller-based development board and its software development framework
- Designed for people without engineering background
- Originally based on Atmel AVR

- Various processors (including ESP32) are now supported. From the software point of view, Arduino provides a common easy-to-develop abstraction layer for embedded systems.

using Arduino library

```
#include <Arduino.h>

#define LED_PIN 21

void setup() {
    pinMode(LED_PIN, OUTPUT);
}

void loop() {
    digitalWrite(LED_PIN, HIGH);
    delay(500);
    digitalWrite(LED_PIN, LOW);
    delay(500);
}
```

without using Arduino library

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"

#define LED_PIN 21

void blink_task(void *pvParameter) {
    gpio_pad_select_gpio(LED_PIN);
    gpio_set_direction(LED_PIN, GPIO_MODE_OUTPUT);
    for (;;) {
        gpio_set_level(LED_PIN, 0);
        vTaskDelay(500 / portTICK_PERIOD_MS);
        gpio_set_level(LED_PIN, 1);
        vTaskDelay(500 / portTICK_PERIOD_MS);
    }
}

void app_main() {
    xTaskCreate(&blink_task, "blink_task",
                configMINIMAL_STACK_SIZE, NULL, 5, NULL);
}
```

Basic Code Structure

Arduino code template

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

The application works like

```
int main() {
    setup();
    for (;;) loop();
}
```

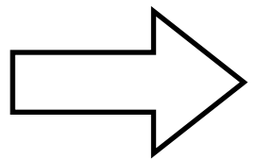
Sketch (1/2)

- A program for Arduino is called a sketch.
  - File extension: ino
- A sketch looks like a C/C++ program

```
void setup() { ... }

void loop() {
  foo();
  delay(500);
}

void foo() { ... }
```



```
#include <Arduino.h>

void foo();

void setup() { ... }

void loop() {
  foo();
  delay(500);
}

void foo() { ... }

int main() {
  setup();
  for (;;) loop();
}
```

Sketch (2/2)

- A platform-specific C++ compiler is used to compile the sketch.
  - A C++ program is generated from the sketch by inserting the default header file (Arduino.h) inclusion, function prototypes, and a starting function.
- You may use some C/C++ files along with the `main' sketch file (.ino).
  - You can only use ".ino" for the main sketch
    - Other files should be written as valid C/C++ code
  - You cannot omit include directives for files other than the default header file (Arduino.h).

Note on Writing Sketches

- I strongly recommend to write your sketches as valid C++ code even if you use Arduino-IDE.
  - Easy to move to other IDEs
    - Some IDE (ex. ESP-IDF) does not support '.ino'.
  - Better editor support.

```
void setup() { ... }

void loop() {
  foo();
  delay(500);
}

void foo() { ... }
```

better

```
#include <Arduino.h>

void foo();

void setup() { ... }

void loop() {
  foo();
  delay(500);
}

void foo() { ... }
```

Note: You don't need to define the starting (main) function as it deeply depends on the platform you use.

The 'main' function in Arduino-ESP32

```
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "Arduino.h"

...

void loopTask(void *pvParameters) {
    setup();
    for (;;) {
        loop();
    }
}

extern "C" void app_main() {
    initArduino();
    xTaskCreatePinnedToCore(loopTask, "loopTask", 8192, NULL,
                            1, NULL, ARDUINO_RUNNING_CORE);
}
```

Example Applications

- M5Stack Samples
  - https://github.com/titech-aos/M5Samples_Arduino
  - https://github.com/titech-aos/M5Samples_ESP-IDF
  - https://github.com/titech-aos/M5Samples_PIO
- How to run (w/ Arduino IDE)
  - 1. clone git repository

    ```
    git clone https://github.com/titech-aos/M5Samples_PIO.git
    ```
  - 2. open one of '.ino' file with Arduino IDE
  - 3. compile and upload

Blink

Repeatedly blinks an LED

```
#include <Arduino.h>

// This example uses GPIO 21 for an LED because the pin is in a
// physically convenient position in M5Stack Core. However, the
// pin is usually used as the SDA of I2C in ESP32. So you should
// make sure that nothing is connected to the I2C (Grove) port
// before running this code.
#define LED_PIN 21

void setup() {
    pinMode(LED_PIN, OUTPUT); // configures LED_PIN as an output
}

void loop() {
    digitalWrite(LED_PIN, HIGH); // turns the LED on
    delay(500);                  // waits for 500ms
    digitalWrite(LED_PIN, LOW);  // turns the LED off
    delay(500);                  // waits for 500ms
}
```

GPIO

- General Purpose Input / Output
- Usually referred as an I/O pin (having no predefined purpose) on a processor (SoC or CPU) or on an I/O component
- Can be configured to have several modes
  - digital I/O, analog I/O, PWM, I^2C, SPI, UART, etc.
  - capabilities
    - input / output
    - pull-up / pull-down / tristate
    - enabled / disabled
    - IRQ

GPIO in ESP32

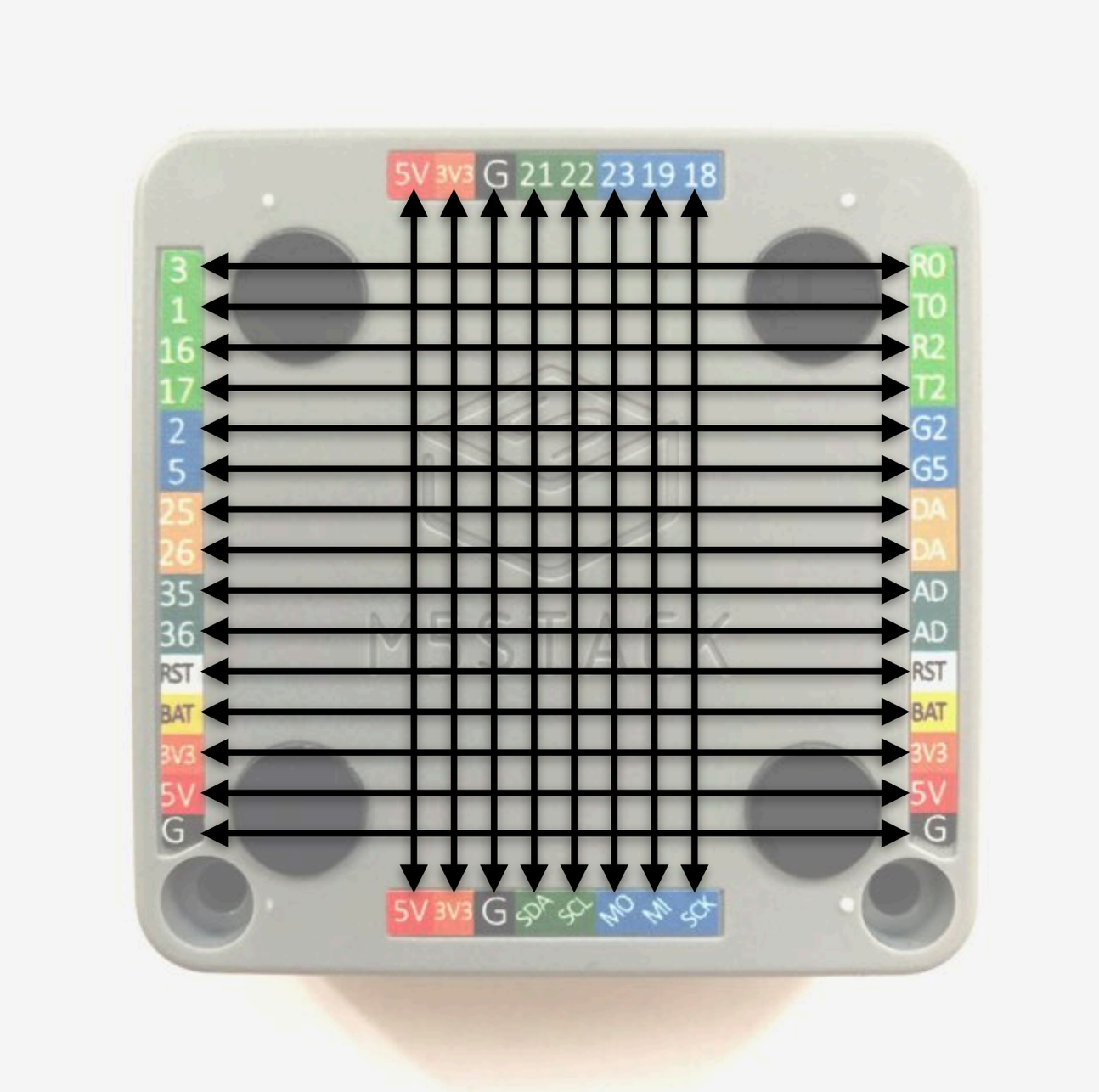
- The ESP32 chip provides I/O multiplexing mechanism that enables to assign one of several I/O capabilities to a specific I/O pin.
  - GPIO (digital input / output)
  - ADC (analog input)
  - DAC (analog output)
  - LEDC (PWM)
  - I^2C
  - SPI
- Documents
  - ESP32 Series Datasheet
  - ESP32 Technical Reference Manual

GPIO in M5Stack (MBUS)

GPIO TYPE	Analog Function	M-BUS				Analog Function	GPIO TYPE
		GND		ADC	G35	ADC1_CH7	I
		GND		ADC	G36	ADC1_CH0	I
		GND		RST	EN		
I/O/T		G23	MOSI	DAC/SPK	G25	ADC2_CH8	I/O/T
I/O/T		G19	MISO	DAC	G26	ADC2_CH9	I/O/T
I/O/T		G18	SCK	3.3V			
I/O/T		G3	RXD1	TXD1	G1		I/O/T
I/O/T		G16	RXD2	TXD2	G17		I/O/T
I/O/T		G21	SDA	SCL	G22		I/O/T
I/O/T	ADC2_CH2/T2	G2	GPIO	GPIO	G5		I/O/T
I/O/T	ADC2_CH5	G12	IIS_SK	IIS_WS	G13	ADC2_CH4/T4	I/O/T
I/O/T	ADC2_CH3/T3	G15	IIS_OUT	IIS_MK	G0	ADC2_CH1/T1	I/O/T
		HPWR		IIS_IN	G34	ADC1_CH6	I
		HPWR		5V			
		HPWR		BATTERY			

https://github.com/m5stack/M5Stack

GPIO Pins

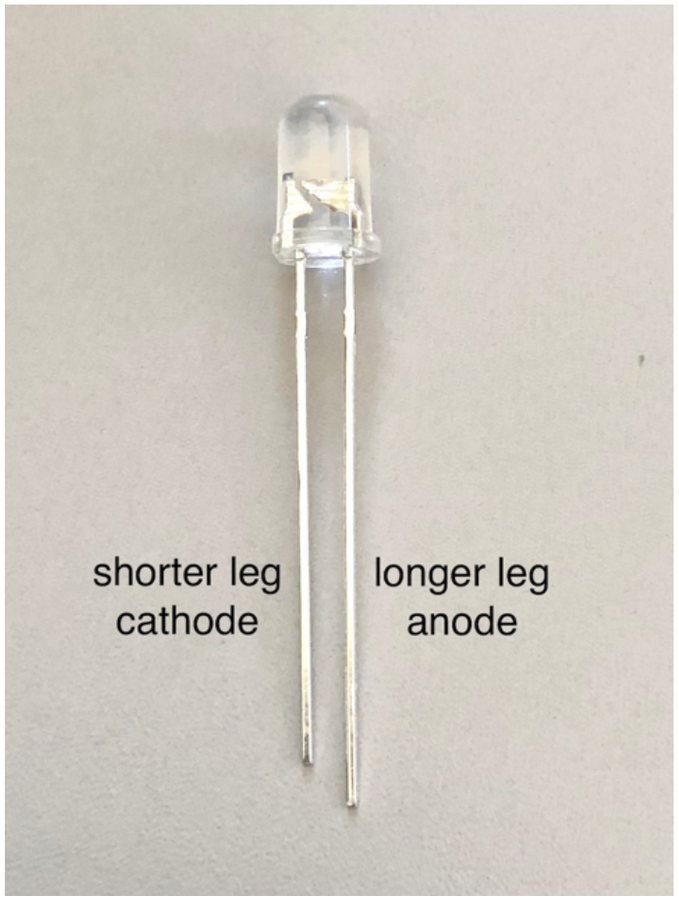


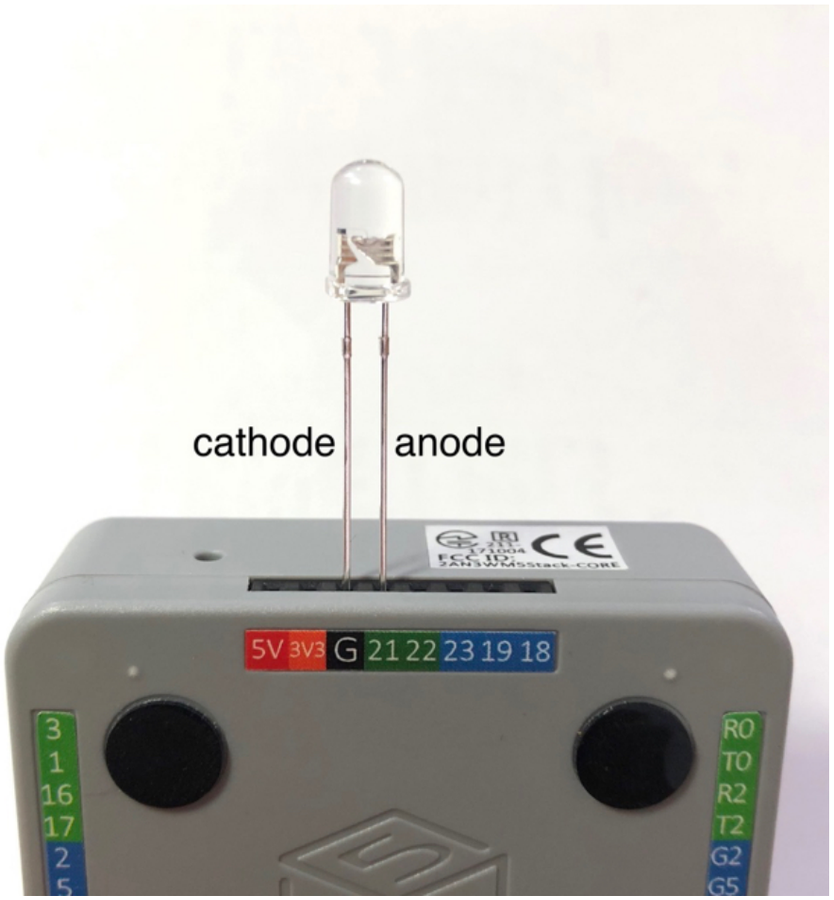
Each corresponding pair of the pins on opposite sides is connected.
ex) GPIO21 = SDA

Pins in the Grove port are connected to SDA, SCL, 5V and G.

How to connect an LED

If you do this, make sure that nothing is connected to the Grove port.





LEDs in the kit have built-in resistors. So you can directly connect them to the GPIO pins. In this example, shorter and longer legs of an LED are respectively inserted into the ground (G) and GPIO 21 holes.

GPIO API

- **void** `pinMode(uint8_t pin, uint8_t mode)`
  - Sets the mode of the pin
  - pin : GPIO pin number
    - 21, G21, etc.
  - mode :
    - INPUT, OUTPUT, INPUT_PULLUP, etc.
- **void** `digitalWrite(uint8_t pin, uint8_t val)`
  - If val is nonzero, output '1' to the specified pin
  - Otherwise (val is 0), output '0'
- **int** `digitalRead(uint8_t pin)`
  - If the pin is configured as INPUT, returns the current value of the pin ('0' or '1'). If the pin is OUTPUT, returns the last output value.

Blink'

```
#include <Arduino.h>

#define LED_PIN 21

void setup() {
    pinMode(LED_PIN, OUTPUT);   // configures LED_PIN as an output
    digitalWrite(LED_PIN, LOW); // set LED_PIN to 0
}

void loop() {
    // toggles the pin value
    digitalWrite(LED_PIN, !digitalRead(LED_PIN));
    delay(500);                 // waits for 500ms
}
```

Notes on GPIO

- Output
  - digitalOut(pin, 0) and digitalOut(pin, 1) output 0V and 3.3V to the specified pin respectively.
- Input
  - You should use 3.3V as '1' for input.
  - ESP32 GPIO pins are NOT 5V tolerable. So you must not connect devices that outputs 5V as '1'.

Time Related Functions

- **`void`** `delay(uint32_t ms)`
  - stops the program (= current task) for specified time period (milliseconds)
- **`unsigned long`** `millis()`
- **`unsigned long`** `micros()`
  - returns the time since the program starts

WifiScan

```
#include <Arduino.h>
#include <WiFi.h>

void setup() {
    Serial.begin(115200);
    WiFi.mode(WIFI_STA);
    WiFi.disconnect();
}

void loop() {
    Serial.println();
    Serial.print("Scanning ... ");
    int n = WiFi.scanNetworks();
    Serial.println("done.");
    for (int i = 0; i < n; i++) {
        Serial.print(WiFi.BSSIDstr(i));
        Serial.print(" (");
        Serial.print(WiFi.RSSI(i));
        Serial.print(") ");
        Serial.println(WiFi.SSID(i));
    }
    delay(5000);
}
```

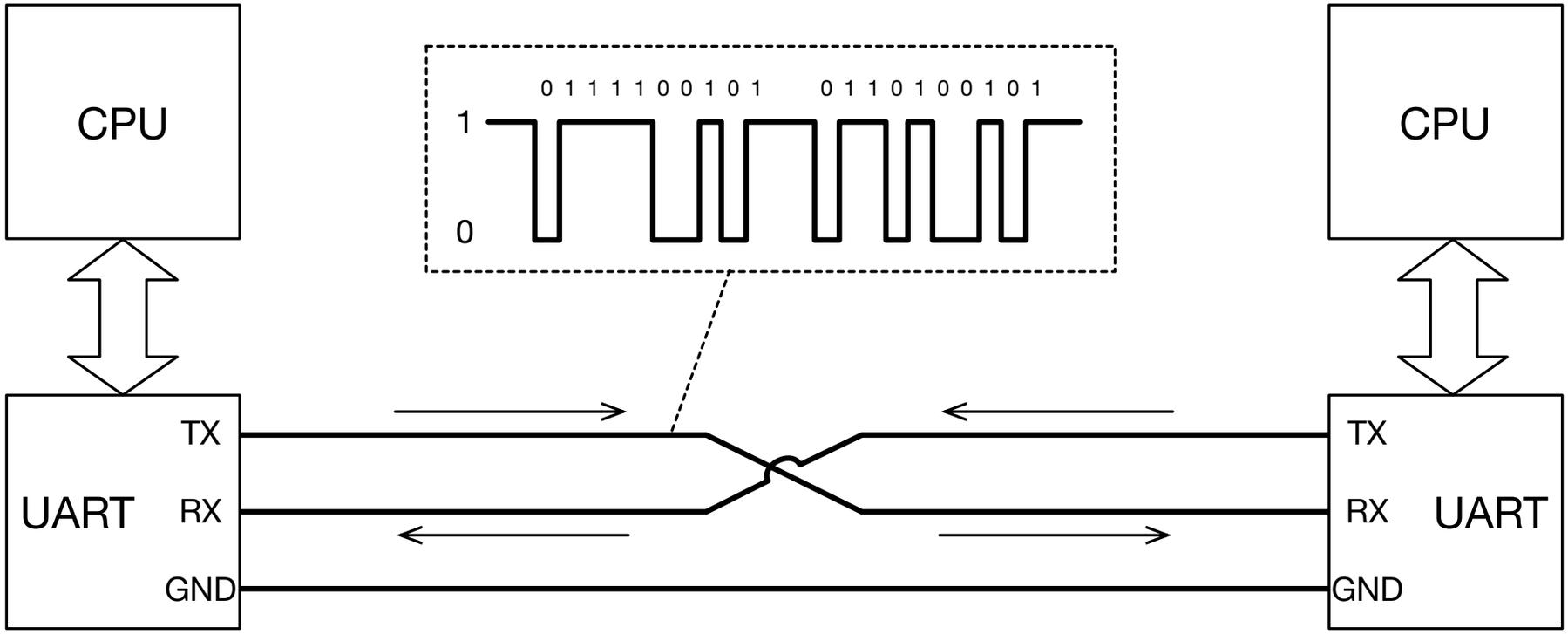
USB-Serial Port

- A virtual serial (UART) port is available via USB
- When you connect your PC to M5Stack, you should find a serial communication port in your PC
  - Mac) /dev/cu.SLAB_USBtoUART
  - Linux) /dev/ttyUSB0
  - Windows) COM?

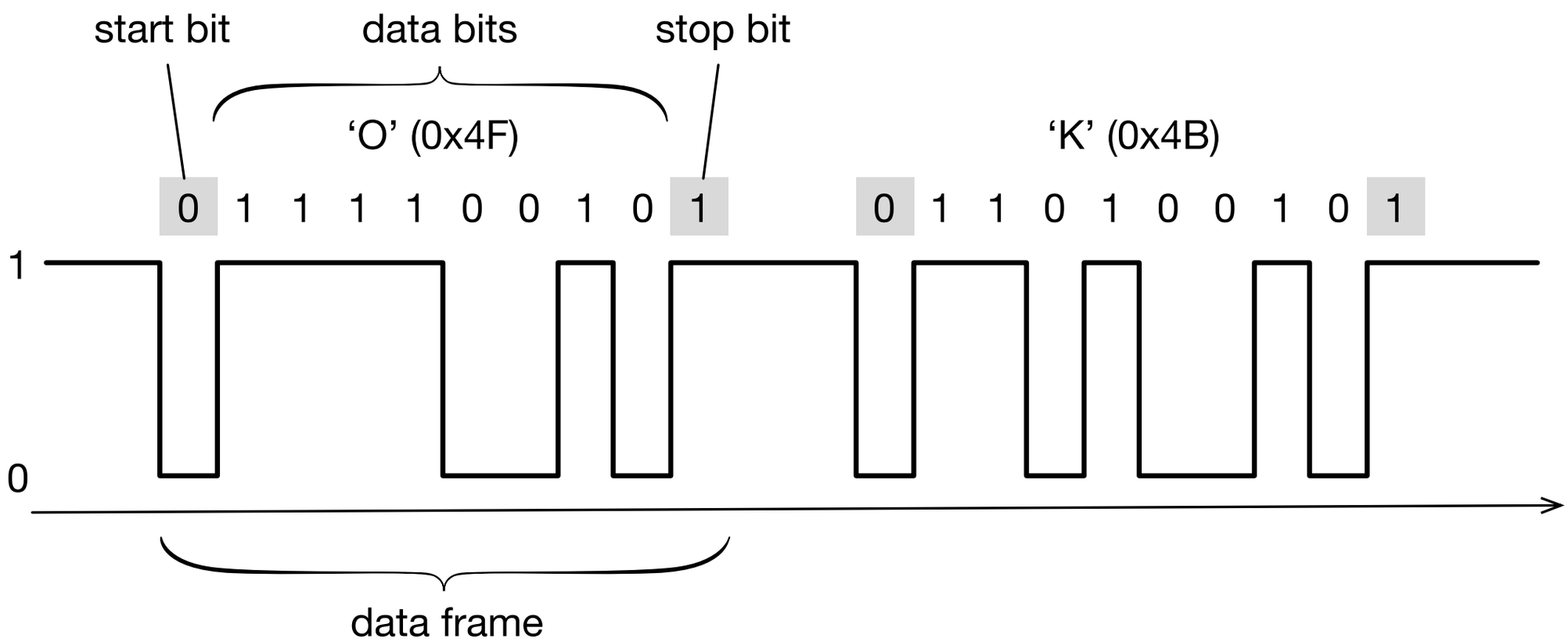
- Serial : default (USB-serial) serial port
  - begin(speed)
  - print, println, printf

UART

- Universal Asynchronous Receiver/Transmitter
- A (legacy) serial communication interface
- Binary bits correspond to voltage levels



UART Signal Format



M5Stack Library

- A library for using M5Stack H/W components
  - https://github.com/m5stack/M5Stack

- M5
  - The pre-defined object that manages M5Stack hardware components
  - M5.Lcd
  - M5.BtnA, M5.BtnB, M5.BtnC
  - M5.Speaker
- Using the library
  - Use M5Stack.h instead of Arduino.h

HelloM5

Greeting Message on LCD


```
#include <M5Stack.h>

const String text = "Hello M5";
const uint8_t text_font = 4;
int mx, my;

void setup() {
    M5.begin();
    M5.Lcd.setTextFont(text_font);
    M5.Lcd.setTextColor(TFT_YELLOW);
    mx = M5.Lcd.width() - M5.Lcd.textWidth(text);
    my = M5.Lcd.height() - M5.Lcd.fontHeight(text_font);
}

void loop() {
    M5.Lcd.fillScreen(TFT_BLACK);
    M5.Lcd.drawString(text, random(mx), random(my));
    delay(1000);
    M5.update();
}
```

Counter


```
#include <M5Stack.h>

int count = 0;
int sw, sh;

void drawCount() { ... }
void setup() { ... }

void loop() {
    if (M5.BtnA.wasPressed()) {
        count = count > 0 ? count - 1 : count;
        drawCount();
    }
    else if (M5.BtnB.wasPressed()) {
        count = 0;
        drawCount();
    }
    else if (M5.BtnC.wasPressed()) {
        count++;
        drawCount();
    }
    M5.update();
}
```

Summary

- Writing M5Stack programs as Arduino code
  - Arduino API
    - GPIO
    - Serial Communication
  - M5Stack specific API
    - LCD
    - Buttons