Introduction to Design (2)

Microcontrollers and Interfacing

Week 02

Serial Monitor, Basic Analogue Input, Basic Digital Output

KUAS 京都先端科学大学
KYOTO UNIVERSITY of ADVANCED SCIENCE

Department of Mechanical and Electrical System Engineering

this week

using the serial monitor

analogue input

digital output (again)

using analogue input to control digital output

using the serial plotter

connecting to host applications

review: sketches

programs (‘sketches’) have two parts:

`setup()` is called once at the start of execution

`loop()` is called repeatedly during execution

```
void setup()
{
  // perform one-time
  // configuration here
  ...
}
void loop()
{
  // repeatedly perform
  // application here
  ...
}
```

`setup` code describes what the hardware *is*

- input/output pins are *configured*

`loop` code describes what the hardware *does*

- input pins are read
- the application logic (simulated hardware) is performed
- output pins are written

(digital) pin 13 has a LED connected to it

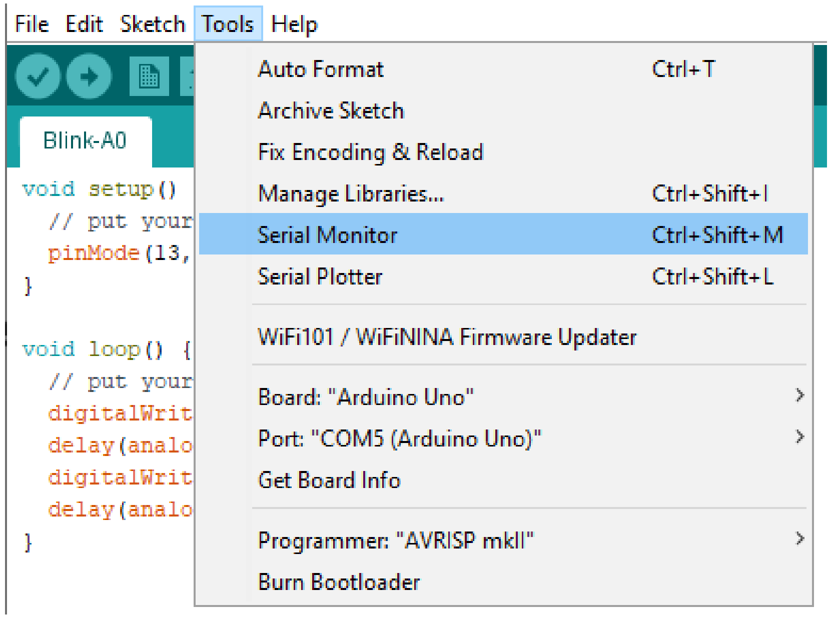
- flashing this LED is sometimes the only form of debugging feedback available

using the serial monitor

the Arduino has a USB serial interface, used for

- programming the hardware (uploading new programs)
- limited user interaction
  - exchange text characters between microcontroller and host
  - better than a flashing LED for debugging

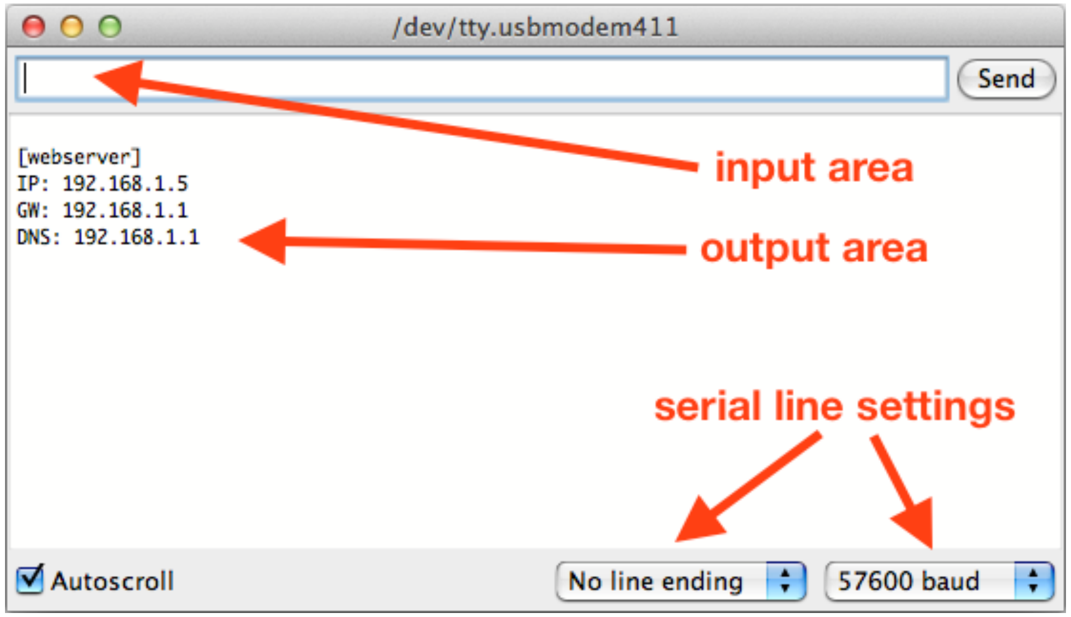
the menu item ‘Tools > Serial Monitor’ opens the serial monitor window:


File Edit Sketch Tools Help
Blink-A0
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port: "COM5 (Arduino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader

```
void setup()
  // put your
  pinMode(13,
}

void loop() {
  // put your
  digitalWrit
  delay(analo
  digitalWrit
  delay(analo
}
```

$\Rightarrow$



```
[webserver]
IP: 192.168.1.5
GW: 192.168.1.1
DNS: 192.168.1.1
```

serial monitor initialisation

USB communication is provided by a collection of functions called `Serial`

- you access the functions as `Serial.`*functionName*`(`*parameters...*`)`

in `setup()` add this line to start serial communication (baud = bits per second)

```
void setup()
{
  Serial.begin(9600); // start serial I/O running at 9600 baud
}
```

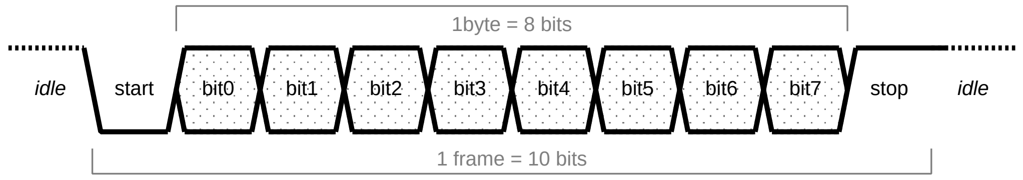
`print()` and `println()` send information over USB to the host computer

```
void loop()
{
  static int counter = 0;
  Serial.print("Hello world "); // send a string to host over USB interface
  Serial.println(counter);      // print counter and a newline on the host
  counter += 1;
  delay(250);                   // limit the rate of characters sent
}
```

`print()` and `println()` can print `int` values, `float` values, `"`strings`"`

serial limitations

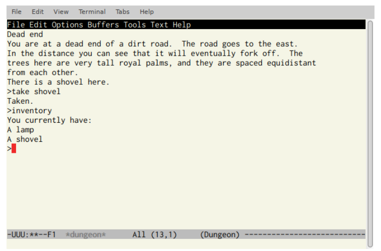
be careful not to send data faster than the configured speed can transmit



one character is $\approx$10 bits (1 start bit, 8 data bits, 0 parity bits, 1 stop bit)

- 9600 baud (the default) is approximately 960 characters per second
- 115200 baud (the fastest) is approximately 11520 characters per second

a *small* terminal screen (80 columns $\times$ 24 lines) is 1920 characters


File Edit View Terminal Tabs Help
File Edit Options Buffers Tools Text Help

```
Dead end
You are at a dead end of a dirt road.  The road goes to the east.
In the distance you can see that it will eventually fork off.  The
trees here are very tall royal palms, and they are spaced equidistant
from each other.
There is a shovel here.
>take shovel
Taken.
>inventory
You currently have:
A lamp
A shovel
>
```

```
-UUU:**--F1  *dungeon*      All (13,1)     (Dungeon) -------------------------
```

the serial buffer (holding data waiting to be sent) is 64 bytes long

- when it is full, your sketch will block (stop running) until space is available

use `delay()` and/or `Serial.flush()` to ensure the buffer is never full

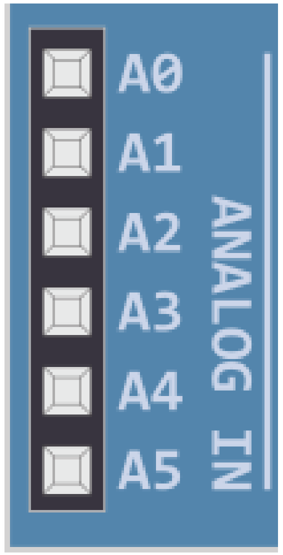
(see reference material in lab worksheet)

analogue input

we can easily generate some more interesting data for `Serial.print()`

there are six dedicated analogue input pins

- named `A0` to `A5`
- values are read with `analogRead(`*`pinNumber`*`)`
- ten-bit analogue-to-digital conversion
  - when input pin is at 0V, *analogRead()* returns 0
  - when input pin is at 5V, *analogRead()* returns 1023



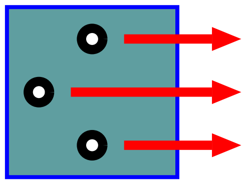
analogue input: creating a variable voltage

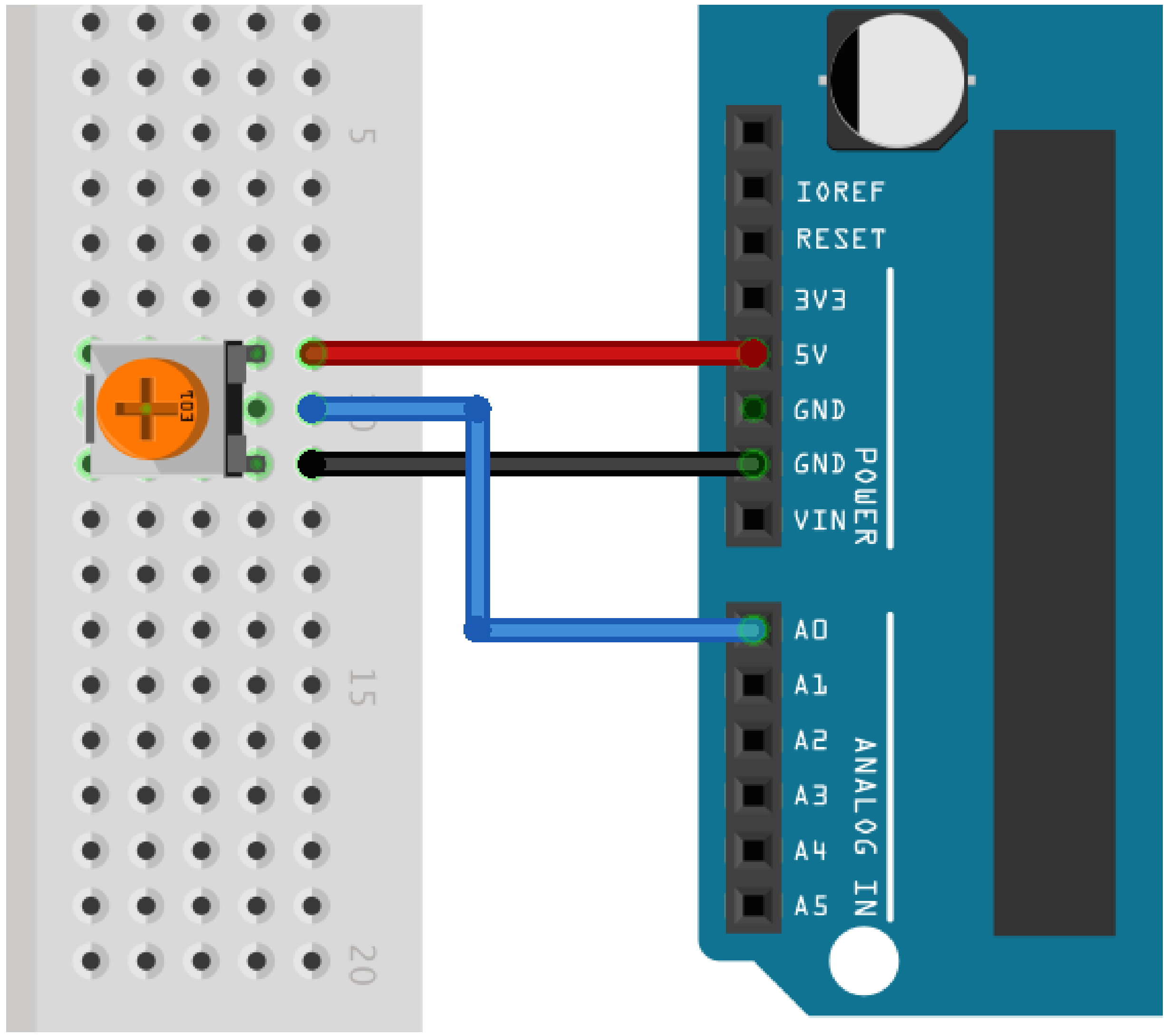
Be *very* careful! Incorrect orientation of the potentiometer can damage it!

Potentiometer pins are fragile. Take great care not to bend the pins while inserting it.

Use minimal pressure and rock it back and forth instead of trying to force it into the breadboard.

bottom view





the *middle* pin of the potentiometer connects to `A0`
never connect the middle pin to `5V` or `GND`
the *outer* pins connect to `5V` and `GND`

analogue input: reading the voltage level

```
void setup()
{
  Serial.begin(9600);                // start USB communication
}

void loop()
{
  int reading = analogRead(A0);      // read voltage on pin A0
  Serial.println(reading);           // send to host via USB
  delay(250);                        // limit transmission rate
}
```

- ‘`int reading;`’ creates an integer variable called `reading`.
- ‘`analogRead(A0)`’ reads the value of the analogue voltage connected to `A0`.
- ‘`reading = analogRead(A0);`’ sets `reading` to that value.
- `int reading = analogRead(A0);`’ both creates and sets the variable.

experiments

modulation is when one thing influences another

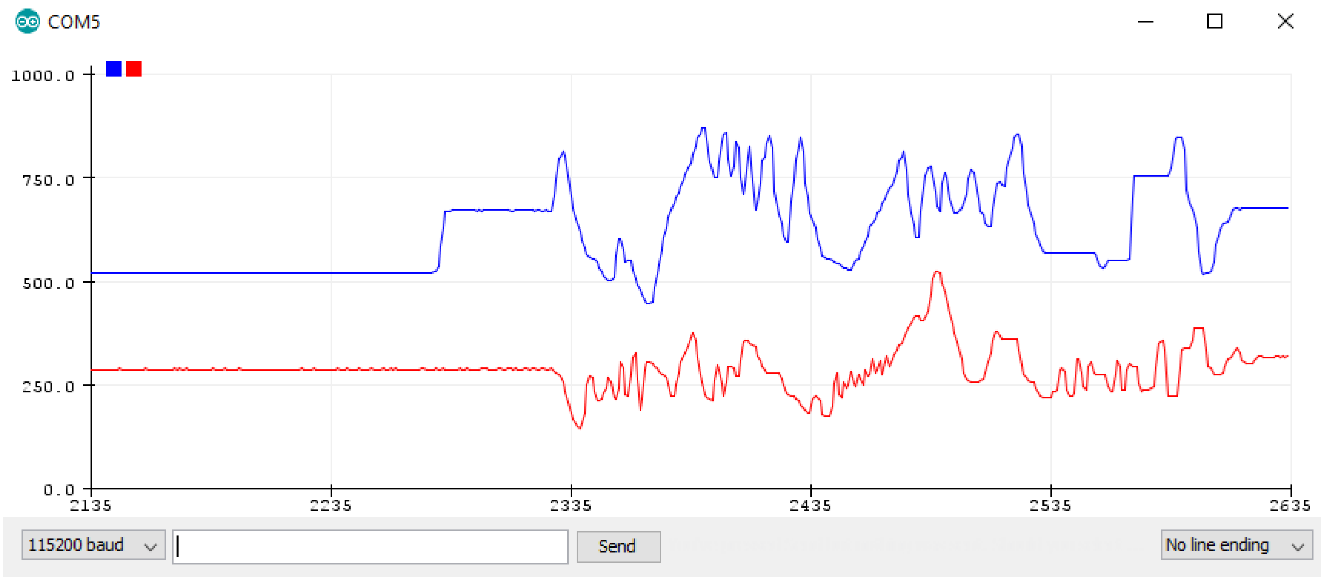
- e.g., one signal affecting another signal or process

we can use the analogue input voltage to modulate the flashing rate of the LED

as well as a serial monitor there is a serial *plotter*

- numbers printed on `Serial` are plotted as a graph

we can display the history of analogue voltages read from `A0`, `A1`, . . .


COM5
1000.0
750.0
500.0
250.0
0.0
2135
2235
2335
2435
2535
2635
115200 baud
Send
No line ending