**Input/output pins on the Arduino**

ENGR 40M lecture notes — July 14, 2017
Chuan-Zheng Lee, Stanford University

*Note: An extended version of this handout is in the spring 2017 notes on the class website.*

An input/output pin, or I/O pin, is the interface between a microcontroller and another circuit. In the Arduino, you configure whether a pin is an input or output using the `pinMode()` function.

### Output pins

An output pin provides $V_{DD}$ or 0 V, by making a connection to $V_{DD}$ or ground via a transistor. You set its state to HIGH (for $V_{DD}$) or LOW (for 0 V) using the `digitalWrite()` function. A (simplified) schematic of an output pin is shown below. You might notice that it looks a bit like a CMOS inverter (or rather, two).

![Output pin schematic](image)

The transistors in the output pin have non-negligible on resistance, so aren’t suitable for driving large loads. When talking about this resistance in relation to an output pin, we call it the output resistance of the pin—in other words, the resistance “seen” by a device connected to the pin. Since this resistance might depend on the state of the pin (HIGH or LOW), it actually has two output resistances. You’ll measure both in prelab 2b.

### Input pins

An *input pin* reads the voltage on the pin as if it were a voltmeter, and returns either HIGH (1) in software if the voltage is close to $V_{DD}$, or LOW (0) if it is close to 0 V. An input pin can be read using the `digitalRead()` function.

The value returned by `digitalRead()` is unpredictable (i.e., could be either HIGH or LOW) when the input voltage is not close to either $V_{DD}$ or 0 V. The precise meaning of “close” varies between microcontrollers, but for the Adafruit Metro Mini\(^1\) in our circuit, the input pin voltage needs to be at least 0.6$V_{DD}$ to qualify as HIGH, and at most 0.3$V_{DD}$ to qualify as LOW.

Switches control connections, not voltage, so if we wish to connect a switch to an input pin, we need a (simple) circuit to “convert” the state of the switch to a voltage. A common circuit used to do this is shown below on the left. The resistor $R_{pu}$ is often called a pull-up resistor, because its function is to “pull up” the voltage of the pin to $V_{DD}$ when the voltage isn’t being driven to ground by the closed switch.

---

In fact, this pattern is so common that microcontrollers provide internal pull-up resistors that you can enable in software by using `pinMode(pin, INPUT_PULLUP)` (where `pin` is your pin number). If you do this, then you can connect just a switch, as shown below right.

As you might guess, the internal pull-up resistor itself is enabled using a PMOS transistor. Thus, here is a (simplified) schematic of an input pin:

**Quick reference**

*Note: In your code, we expect you to use sensible variable names and to define constants for pin numbers.*

Lines that should go in your `setup()` function:

```cpp
pinMode(pin, INPUT);  // configures pin pin as input without pull-up
pinMode(pin, INPUT_PULLUP);  // configures pin pin as input with pull-up
pinMode(pin, OUTPUT);  // configures pin pin as output
```

Lines that control or read the pins:

```cpp
byte x = digitalRead(pin);  // reads the value of pin pin
digitalWrite(pin, HIGH);  // sets the state of pin pin to HIGH
digitalWrite(pin, LOW);  // sets the state of pin pin to LOW
```