Section 4 Cheat Sheet

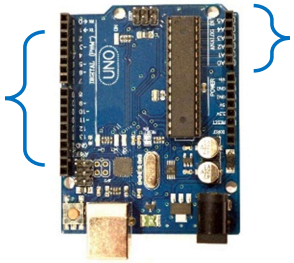
Digital Pin Modes

Digital Pin: **INPUT** Mode	Code:	`pinMode(myPin,INPUT);`
		`pinMode(myPin,INPUT_PULLUP);`
		`digitalRead(myPin);`

	Used for:	• Receiving information (0 or 1)

Digital Pin, **OUTPUT** Mode	Code:	`pinMode(myPin,OUTPUT);`
		`digitalWrite(myPin,HIGH);`
		`digitalWrite(myPin,LOW);`
		`analogWrite(myPin,#);`
		(# is an integer from 0-255,
		0=0% duty cycle, or LOW
		127=50% duty cycle, or HIGH half the time
		255=100% duty cycle, or HIGH all the time)

	Used for:	• Sending information (0 or 1) • Powering low-power devices (<0.2 W) • Serial communications (e.g. I^2C) • PWM (Pulse-Width Modulation), pins 3,9,10,11: 490 Hz, pins 5,6: 980 Hz.



Analog Pin Modes

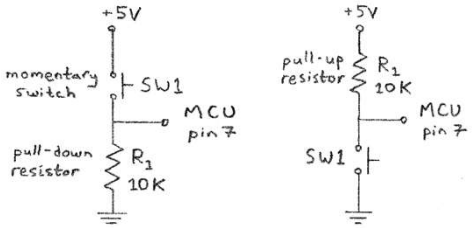
Analog Pin, **INPUT** Mode	Code:	• No need to declare `pinMode()`
		`analogRead(myAnalogPin);`
		• To use the internal 1.1V analog reference:
		`analogReference(INTERNAL);`
		• To use an external analog reference:
		`analogReference(EXTERNAL);`
		• Remember to plug a voltage into the AREF pin to use an external analog reference.
		• `pinMode(myPin,INPUT_PULLUP);//optional`

	Used for:	• Receiving a voltage. (0→1023 , maps to 0→5V, or 0→AREF)

Analog Pin, **OUTPUT** Mode	Code:	`pinMode(myAnalogPin,OUTPUT);`
		`digitalWrite(myPin,HIGH);`
		`digitalWrite(myPin,LOW);`

	Used for:	• Same as digital pin in OUTPUT mode (you can use analog pins if you run out of digital pins). • Not PWM-capable.

Pull-up and pull-down resistors:



Converting an analog reading:

```
//to a float:
float volts=analogRead(myPin)*5.0/1023.0;
//to PWM range 0-255:
byte x=analogRead(myPin)/4;
//to new int range (ymin to ymax):
int reading=analogRead(myPin);
int y=map(reading,0,1023,ymin,ymax);
```

Serial Monitor

```
Serial.begin(9600); //start monitor
Serial.print("Hello");
Serial.println("Hello"); //new line after
Serial.println(reading,4); //with 4 decimals
```

void function with no parameters:

```
void doSomething1(){
  actions in here; //nothing returned
}
```

void function with parameters:

```
void doSomething2(byte x, int y){
  //local copies made of x and y
  actions in here; //nothing returned
}
```

int function with parameters:

```
int calcSomething1(byte x, int y){
  actions in here;
  return answer; //return this int
}
```

float function with parameters:

```
float calcSomething2(byte x, int y){
  actions in here;
  return answer; //return this float
}
```

float function with array as parameter:

```
float calcSomething3(int x[], int n){
  actions in here;//pass array size as n
  return answer; //return this float
}
```

To call the functions above:

```
doSomething1();
doSomething2(3,5);
int answer1=calcSomething1(5,6);
float answer2=calcSomething2(4,7);
int arr[3];
float answer3=calcSomething3(arr,3);
```

Call-by-Value

```
void myFn1(int y){
//local copy made of y arg
  y++; //increase copy of y
       //by 1
}
```

Call-by-Reference

```
void myFn2(int &y){
//y arg address passed
  y++; //increase y by 1
}
```

To call the functions above:

```
int x=6;
myFn1(x); //after fn, x=6
myFn2(x); //after fn, x=7
```

#define Statements

```
#define MYPIN 6 //no ; here
#ifdef MYPIN
   do something here;
#else
   do something else;
#endif
// as function:
#define CUBE(x) (x*x*x)
float y=CUBE(3); //y=27
```