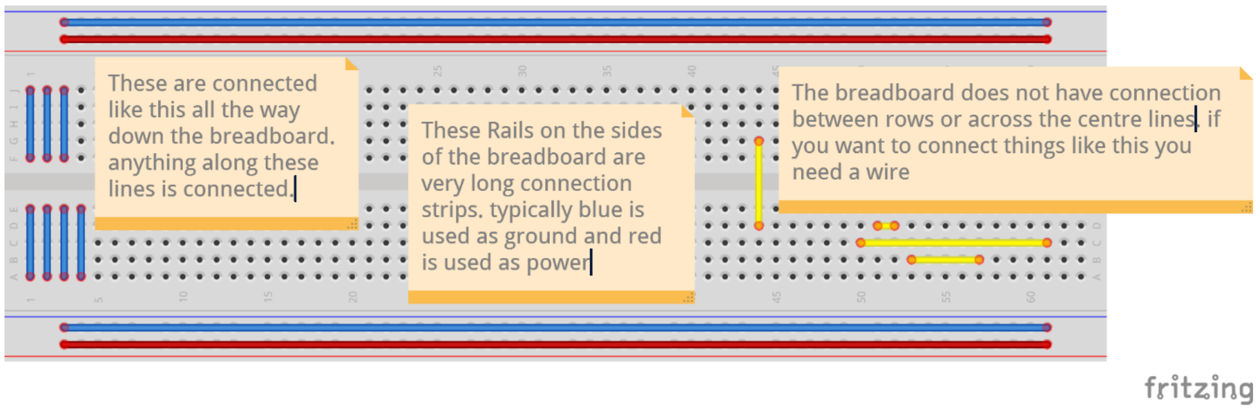
Breadboard Connection Instructions


These are connected like this all the way down the breadboard. anything along these lines is connected.
These Rails on the sides of the breadboard are very long connection strips. typically blue is used as ground and red is used as power
The breadboard does not have connection between rows or across the centre lines. if you want to connect things like this you need a wire
fritzing

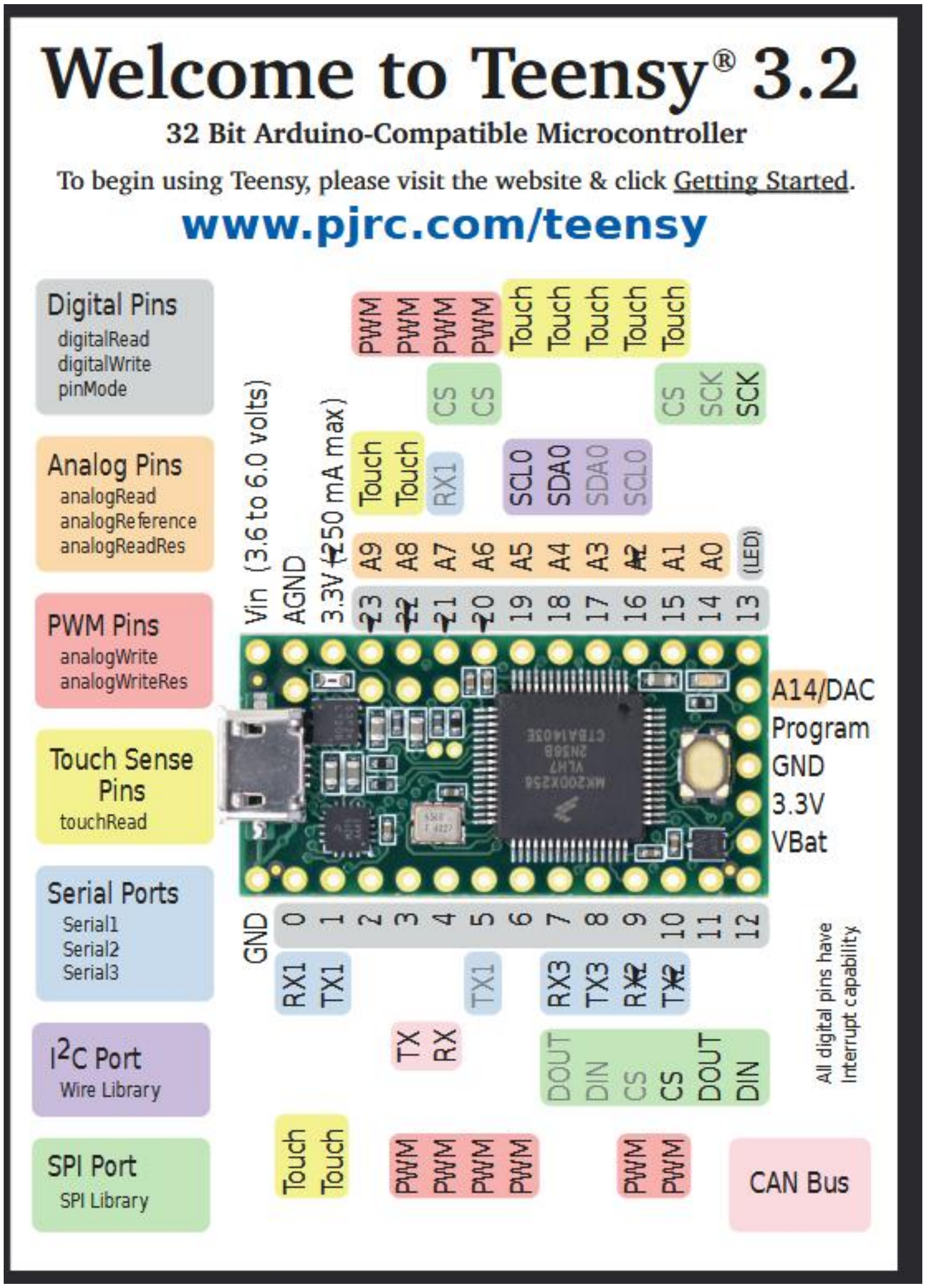
In your Kit

- 1 breadboard (see above picture)
- 2 buttons
- 1 LED
- 1 220 Ohm Resistor
- Assorted jumper wires

Teensy Circuit Instructions

1. Place the teensy on the breadboard so that each row of pins is on a different side of the center gap in the breadboard DO NOT PLUG IT IN YET
2. Connect Teensy pin GND to a long blue rail marked with a (-) sign. This will be your ground. Anything connected to ground will be on this rail.
3. Place the LED on the breadboard so that each leg is on a different connection row. (see breadboard diagram)
4. Connect Teensy Pin 9 to the Anode of the LED (long leg)
5. Connect the Cathode (short pin) of the LED to one side of a 220 ohm resistor Each side of the resistor should be on a different connection row.
6. Connect the other side of the resistor to Ground
7. Place the buttons so that they straddle the center gap of the breadboard. (you may want to put these a good distance apart to make the game easier)
8. Connect Pin 2 to one corner pin of the button

9. Connect the corner button pin diagonal to the previous pin to ground
10. Connect pin 3 to one corner of the other button
11. Connect the corner button pin diagonal to the previous pin to ground
12. Plug in the teensy



Teensy Pinout

Code Reference guide

Operators:

Symbol	Meaning	Example
+	Addition. adds 2 things together	A + B
-	Subtraction. Subtracts term 2 from term 1	A - B
*	Multiplication. Multiplies terms	A * B
/	Division. Divides term 1 by term 2	A / B
=	Assigns a value to a variable	A = 6 (A now means 6 until it is changed)
==	Equals. Compares 2 values and sees if they're the same	A == 6 (the value of A is not changed but if A is 6, this statement is True and False otherwise)
!=	Does not equal. Campares two values and sees if they're different	A != 6 (This statement is true if A is not 6)
>	Greater than. Compares 2 terms and sees if term 1 is greater than term 2	A > 6 (This is true if A is greater than 6)
<	Less than. Compares 2 terms and sees if term 1 is less than term 2	A < 6 (This is true if A is less than 6)
>=	Greater than or equal to	A >= 6 (True if A is greater or equal to 6)
<=	Less than or equal to	A <= 6 (True if A is less than or equal to 6)
&&	Logical AND. Is true only if both statements are true	A >= 2 && A !=4 (True only if A is greater than or equal to 2, AND if A is not 4)
||	Logical OR. is true if one or both of the statements are true	A >= 2 || A != 4 (True if either A is greater than or equal to 2 OR if A is not 4 or if both or true)

++	Increment. Increases the value of the term by one.	A++ (If A was 6 before, It is now 7)
--	Decrement. Decreases the value of the term by one	A -- (If A was 6 before, it is now 5)

Assigning Variables, Data Types and Basic Syntax

int A = 6;

int is the type of the variable. int means that A is an integer. Some other data types include:

float: for decimal numbers.

long: long int. An integer but with more room for data than a regular int for extra big numbers

unsigned int and unsigned long: These types are similar to int and long but they don't use any data to specify the sign (positive or negative)

bool: a Boolean. something that can only have values of true or false

void: Used to specify that a function doesn't return anything, as opposed to int (or other) type functions that will return a value of a specific type.

A is the variable name. This is what we are assigning a value to.

= is the assignment operator. (explained in the operator table.)

6 is the value we are assigning to A.

; all lines must end with a semicolon

Loops:

for loop:

syntax:

```
for(int i=0; i<10; i++){
    //do something

}
```

//do something: Anything written after "//" is a comment, it does not affect the code. Here a comment is used to denote where code that performs a task in the loop should go

in the Parentheses in the for loop, you will notice that there are 3 parts separated by semicolons. First you define a variable with a starting point, (in this case it's an integer “i” starting from 0.) Then you set a limit for how long your for loop will run, (in this case it will run as long as i is less than 10.) Finally you set your variable to increment every run through of the loop. Because you’re setting up all the limits ahead of time a for loop is typically a safer option than a while loop and you avoid risking setting an infinite loop. However you do lose some versatility and a for loop is not always possible or practical in every situation.

while loop:

syntax:

```
while(A != 6){
     //do something
}
```

Here in the parentheses, you only have one part. This is a condition that the loop will run under. Here, as long as A is not equal to 6 the loop will continue. If A becomes 6, the statement will be false and the loop will stop.

void loop()

syntax:

```
void loop(){
     //do something
}
```

Technically this one is actually a function. This is something specific to arduino’s which are generally intended to continue performing a task for long periods of time rather than simply run 1 task and then stop. This is where you would write code for your machine that repeats over and over again.

Functions:

pinMode(pin, mode) this function sets the mode a specific pin will be in. for example:

```
pinMode(8,INPUT);
```

means you are setting pin 8 to receive input.

Serial.begin(baudrate) This function starts the communication with the teensy which allows us to send and receive communication with it through our computer. The baudrate is the signaling speed which we must set to match what the teensy is outputting.

```
Serial.begin(9600);
```

analogRead(pin) This function reads the analog voltage of a specified pin and converts it to a digital value.

```
analogRead(A0);
```

This means that you are reading the voltage from analog pin 0. if that's not connected to anything you will get random electrical noise.

randomSeed(newseed) This function specifies where you want your random sequence to start. Computers aren't very good at making truly random numbers. By calling

```
randomSeed(analogRead(A0));
```

we are varying where the "random" sequence starts with a random noise reading, making the game more unpredictable.

digitalRead(pin) This function returns either HIGH or LOW depending on the voltage of the specified pin.

digitalWrite(pin, value) This function sends a value of either HIGH or LOW to a specified pin. Used to turn on or off components and send signals. The pin mode must be set to OUTPUT for this.

delay(milliseconds) This function tells the computer to wait a specified number of milliseconds before continuing.

random(howsmall, howbig) Returns a random number in the specified range. Even more random if randomSeed is used with an unconnected pin.

millis() returns the current number of milliseconds that have passed since the program started running

Serial.print(stuff) prints stuff out to the user. Put things in quotation marks to print the literal words, no quotation marks for variables. you cannot put variables and text in the same print statement

Serial.println(stuff) prints stuff out to the user on its own line.