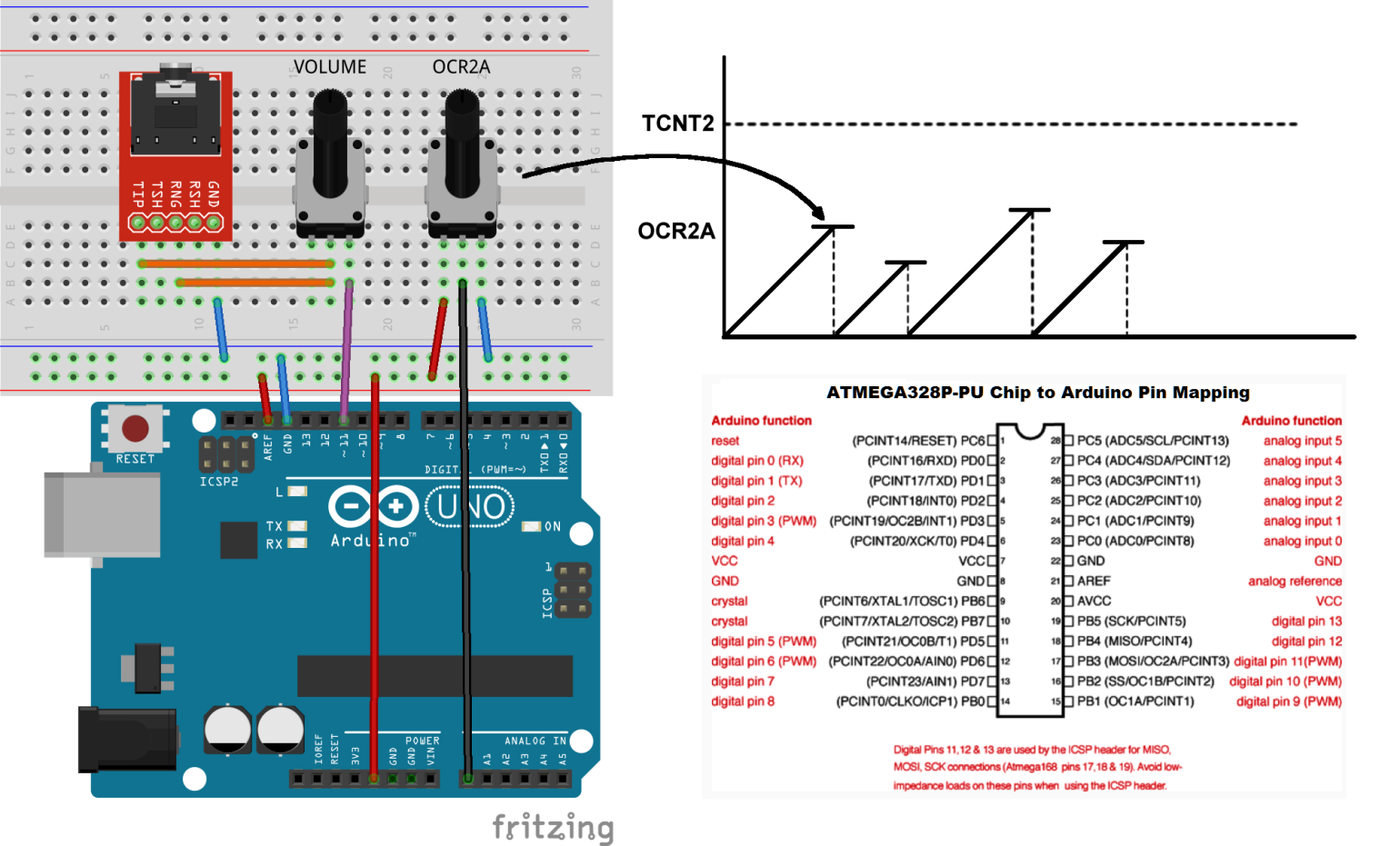
The left half the Fritzing diagram below is familiar to you. The potentiometer marked VOLUME accepts the PWM signal from pin 11 (OC2A) on its C leg, adds a resistance, and then passes it along to the TIP and RNG channels of the audio jack breakout board. You gained familiarity with the technique after coding Timer1’s CTC Mode 4 in a very recent investigation in class in which the counter register, TCNT1, is reset to 0 on a compare match with OCR1A. In this exercise, a second potentiometer is added, not to contribute resistance, but to act as a voltage divider. The analog voltage will be read from the sweep pin on PC0 (A0). Through code you are to develop, this digital value will enable the operator to influence the value of OCR2A, mechanically, on the fly.



You will recall that the Arduino C statement analogRead(pin) invokes the ATmega328P’s analog-to-digital processor and returns a 10-bit binary result that approximates the analog voltage value between 0 and it’s reference value (AREF) which is tied to 5V in the Fritzing image above.

### Task

1. Using either mid-level C code (*maximum register use*) or standalone assembly (*it doesn’t matter which*) develop code on the blue page provided that will implement Timer2’s CTC Mode. This will result on a PWM frequency appearing on OC2A (digital pin 11).

2. At precisely 1-second intervals an analog-to-digital conversion will be started that will acquire an approximation to the voltage on A0. When the conversion is complete the value will be used to update OCR2A. For the purpose of the Challenge do not concern yourself with a 10-bit value being jammed into an 8-bit target (*unless you want to*).

3. With no less than 10 minutes remaining in the period, **neatly transcribe** your working copy (blue) to the light yellow page provided with a reasonable attempt at commenting.

4. **Turn the page for a general description of the analog-to-digital conversion process on the ATmega328P and a list of supporting resources you have provided with in this package**.

### Brief Overview of the Analog-to-Digital Conversion Process

A handful of registers (*outlined at the end of the Chapter 24 pages you have been provided with*) are manipulated to obtain the 10-bit digital approximation. Within these registers you’ll find the strategies for,

a) selecting the desired analog pin

b) disabling the pin for *digital* use to save power

c) setting a prescaler to give the ADC unit a little time to undertake the conversion

d) a flag to enable the unit as a whole

e) a flag to adjust the result to make it more compatible for an 8-bit target register

f) a flag to enable an interrupt to be generated when a conversion is complete

g) a trigger to start a conversion

### Supporting Resources

1. The full Analog-to Digital-Conversion Chapter from the ATmega328P datasheet (pp. 237-252)

2. The Timer1 Register Section from the ATmega328P datasheet (pp. 131-137)

3. The required pages of the Timer2 Register Section from the ATmega328P datasheet (pp. 153-158)

4. The complete list of ATmega328P interrupt names and vectors

5. A blue page to develop your thinking, algorithm, and rough code

6. A light yellow page to neatly transcribe your rough code into clear, legible and easy to follow commented code.

### Submission

At the end of the Challenge period please hand me your named, light yellow solution.

### ATmega328P Vector Table

