Pulse Drive Project

Heart Rate Video Game Controller

By Max Tayler
Document last updated: 2020 May 23

Disclaimer

This adapter for a heart rate sensor is intended for educational and entertainment purposes only. It is not part of a medical device, nor does it claim to be. The common human resting heart rate is 60 beats per minute, down to 50 if you are athletic, up to 100 if you are child. Those beats typically have only minor fluctuations in rhythm. If you have questions about your heart rate or rhythm or any other medical symptoms, you should promptly talk to your doctor or other clinician. Here in British Columbia anyone can call HealthLink at 8-1-1 for free advice from a registered nurse.

Table Of Contents

Introduction

This simple Arduino based controller is designed to convert the Seeed heart rate sensor output into a keyboard signal for my video game, The Chill Challenge. A player wins the Chill Challenge with a steady heartbeat. The game is fun, but it was designed to detect variations in heart rate related to anxiety and get to people into a positive “flow state”.

The actual sensor sends a harmless infrared pulse through your fingertip or earlobe. The detector sees variations caused by blood flow. The little Seeed adapter box turns that analog signal into a clean square wave in time with your heart beat. The Arduino takes that digital signal and translates it to key presses and key releases that it delivers to a computer through a USB port. The game counts and times the pulses and displays your changing pulse rate.

Components

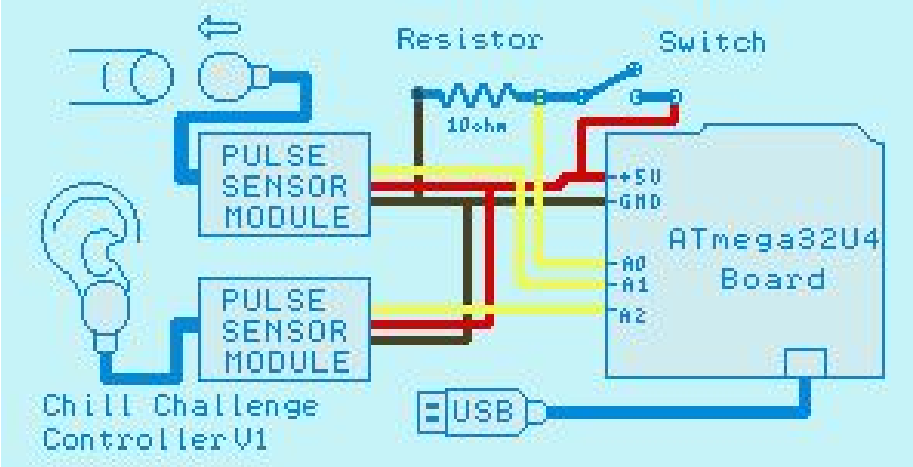
You can use an Arduino board with headers for quickly testing the concept and the game. You will need the Grove 4-pin converter table to just plug the sensor into the Arduino, no soldering required. The switch and resistor are not essential, but when the controller board is plugged in and running and the ear clip is in place, the board delivers a keystroke with each beat of your heart. If you are in an editor, not the game, you may see your cursor hopping across the screen in time with your pulse.

If you want a controller for glitch-free sustained use you will need an Arduino board with solder pads instead of plug-in headers. You can skip the adapter cable, but you may want to add display LED's with appropriate resistors so that inexperienced users can confirm that the sensors are correctly placed

X 1 Arduino Leonardo -
https://store.arduino.cc/usa/arduino-leonardo-with-headers
X 1 or 2 Pulse Sensor Module -
https://www.seeedstudio.io/Grove-Ear-clip-Heart-Rate-Sensor-p-1116.html

X1 or 2 converter cable 2mm male to 2.54mm male pins for pulse sensor.
https://www.seeedstudio.io/Grove-4-pin-Male-Jumper-to-Grove-4-pin-Conversion-Cable-%285-PCs-per-Pack%29-p-1565.html
X 1 Switch or Potentiometer.
X 1 10K ohm resistor

Schematics



Tips

If you have experience, you can plug everything together on a board with headers, download and install the Arduino development environment, assign the correct port, load the code onto the board, and try out the Chill Challenge in less than twenty minutes. With practice, a teacher can fit everything into one lesson. Someone new to Arduino working on their own will take an evening to get things running, possibly requiring a couple of phone calls to a technical friend. Someone with a little experience might take a whole morning purchasing a cute little plastic box and figuring how to fit everything into it, then spend an afternoon fiddling the bits into place with double-sided tape and hot glue and getting the wires the right lengths and protecting everything with heat shrink. The actual soldering is a matter of only a few seconds here and there if you have a nice temperature controlled precision soldering iron.

Code

Links to a web editor and downloadable version of the IDE are located here
https://www.arduino.cc/en/Main/Software

When you launch the Arduino IDE, be sure to use the commands under “Tools” to correctly identify the board you are using and what port your system prefers. Before loading the code for the controller for make sure all your components are tested and working. First test to see if the Arduino board and IDE communicating correctly, by loading the blink program. You can find the blink program in the IDE located under File>Examples>Basics>Blink. There are also many tutorials online to help you out with this.

Next, test to see if the sensor works using the Analog Input program located File>Examples>Analog>AnalogInput. You will need to change the sensor pin to be read from A0 to A1 in line 30.

The current version of the IDE creates an Arduino folder in the users Documents, the best place to locate the code for the game.