

Optical Snail Heart Rate Monitor

Quentin Geissmann

Monitoring physiological variables is crucial in many areas of biology. It is particularly difficult to study the physiology of small animals whilst minimising experimental interference or under natural conditions.

Land snails are remarkable for their ability to enter in dormancy states when the weather becomes too cold (hibernation) or too dry (estivation). They can remain several months in these respective states as their metabolism dramatically slows down, which reduces their need for food and water.

Recording heart rate of snails could reveal a valuable tool to shed light on the mechanisms underlying these exceptional phenomena. In addition, it could help to understand temporal patterns of activity and, for instance, foraging and mating strategies.

Garden snails (*Helix aspersa*) are fairly large terrestrial invertebrates (more than 10 g for adults) capable of pulling their own weight several times. In addition, their shell is a stable mineral support where small circuits can simply be glued with, presumably, little inconvenience for the animal. It is possible to observe the heart of young snails, through their thin shell with naked eyes. In adult snails, the same observation can be achieved by casting strong light through the shell. It should therefore be possible to measure light attenuation over time in order to infer heart rate.

For these reasons, land snails appear particularly suited to carrying small electronic equipment in order to record an important physiological variable for extended periods of time. This project describes an inexpensive and non-invasive device to monitor heart rate in land snails.

Project Aim

The aim of this project is to build a small device that can monitor a snail's heart rate and is small enough to be carried by a garden snail (*Helix aspersa*). When sending red light through the bottom of the shell, the moving heart of the snail can be seen. From this observation, the goal is to fit a low power red LED on one side of the shell whilst recording the intensity of the transmitted light, using a phototransistor (PT), from the other side.

In order to keep the device lightweight, thin wires will be used to convey analogue signal between the animal and an Arduino Uno board. Then, the serial port will be used to communicate data to a computer which will display the signal in real time using a `python` script.

In order to save energy and reduce the amount of heat generated by the LED, the circuit will be powered only shortly before recording data points, and turned off immediately after. For this purpose, a digital pin will be used.

Techniques Used

- Reading values from a PT
- Using a digital pin to drive power
- Oversampling signal to improve resolution of A/D conversion.
- Real time data visualisation using `python` in combination with `pyserial` and `pygame`

Materials

- Arduino Uno
- Computer
- Low voltage (around 3 V) bright red LED. Ideally, a small, surface mounted, one with a flat lens and a wide ($>100^\circ$) angle.
- Phototransistor (PT) that matches the wavelength of the above LED.
- 68 Ω resistor
- 100 k Ω resistor
- A two rows, straight, PCB header
- A one row PCB socket
- Some thin wire
- Large garden snail
- Superglue
- Blu tack
- A large snail (an adult *Helix aspersa* should be perfect)

@edwbaker -> you will probably want to change the links to your cloned/reorganised version here

@edwbaker -> Maybe you want this as a footnote, or not at all.

The commercial references to some of the electronic components can be found at (https://github.com/gilestrolab/snail_back_pack/blob/master/protocols/rs_numbers.md)

Code Download

@edwbaker -> fix links at some point

- Arduino code (https://github.com/gilestrolab/snail_back_pack/blob/master/arduino_prototypes/s)
- Python code (https://github.com/gilestrolab/snail_back_pack/blob/master/scripts/serial_monitor.py)

Supporting Materials

- Video of the protocol (<http://dx.doi.org/10.6084/m9.figshare.1294198>)

Theory

Land snails have a semi-closed circular system where their oxygenated hemolymph (i.e. blood) is carried from their lung to their heart in a pulmonary vein. Then, the heart, which has two cavities (atrium and ventricle), pumps the blood into the aorta artery. As opposed to a closed circulatory system (e.g. in vertebrates), the hemolymph then mixes with the extracellular fluid before diffusing to the lung.

The heart rates of snails is regulated by oxygen demand and can scale from more to one beat per second (e.g. when the animal is active and the temperature high) to as low as one a minute (e.g. during hibernation). Much like in vertebrates, the electrical activity of the heart can be recorded by electrocardiography (ECG) in order to measure heart rate. However, this technique is quite invasive (i.e. holes need to be drilled in the shell), and technically challenging.

In human, pulse-oximetry can be used to reliably and non-invasively infer heart rate by comparing red and infrared absorbance. In snail, since a difference in light absorbance is visible during heart beats, it should be possible to record these variations using simple electronics.

Light-Emitting Diodes (LEDs) are used in a wide range of applications. They feature low power consumptions and some of them are capable of generating a relatively narrow emission spectrum (e.g. 95% of the emission within a 100 nm band). In this project, we will power an LED with a forward current $I_F = 40$ mA, and a forward voltage $V_F = 2.6$ V from a $V_B = 5$ V board. Therefore, we need to drop $V_R = V_B - V_F = 2.4$ V with a resistor (in series with the LED).

@edwbaker => maths formatting around here?

According to Ohm's law, the resistance is $R = V_R/I_F$. So, in our case, $R = 60$ Ω , which means we should be able to use a 68 Ω resistor.

Phototransistors (PTs) are essentially transistors that allow variable current to pass through as a function of light intensity. As opposed to PTs, they have a relatively fast response time. In order to record light intensity, we can record how much current "leaks" through the PT. To do so, we put a resistor in series with the PT and measure the voltage across it with the analogue pin. The value of the resistance affect the gain and should depend on the type of PT and the range of light intensity. In our case, a 100 k Ω resistor should give satisfying results.

The circuit

The circuit in itself is really simple:

Note that the **digital pin 2 is used instead of the 5 V pin**. This way, we can turn the circuit on and off from the Arduino.

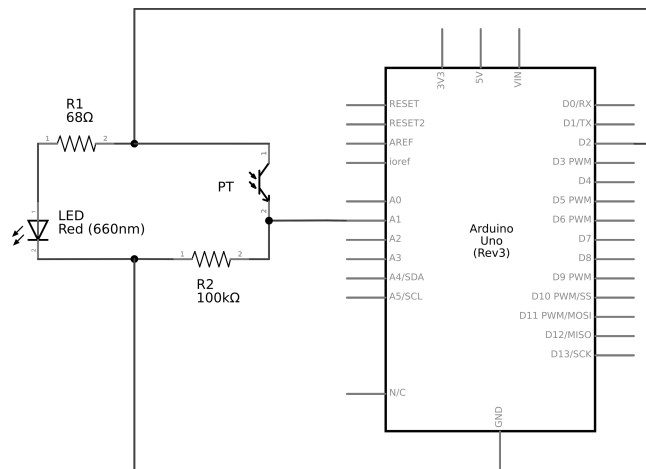


Figure 1: Circuit schematic

Putting it Together

The end goal is to build a sort of saddle that can be glued onto the shell, and then plugged to wires for recording purposes. View from the bottom, the final product could look like this:

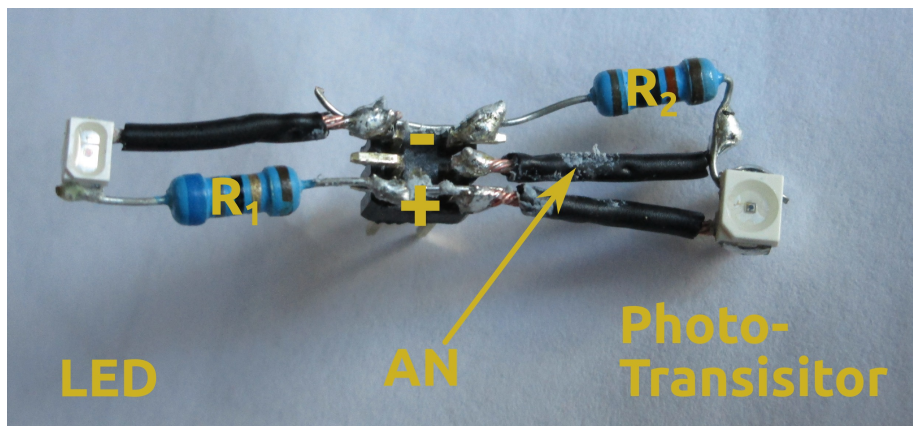


Figure 2: Monitoring device, bottom view

And from the top:

In both figures, 'AN' is for the analogue pin.

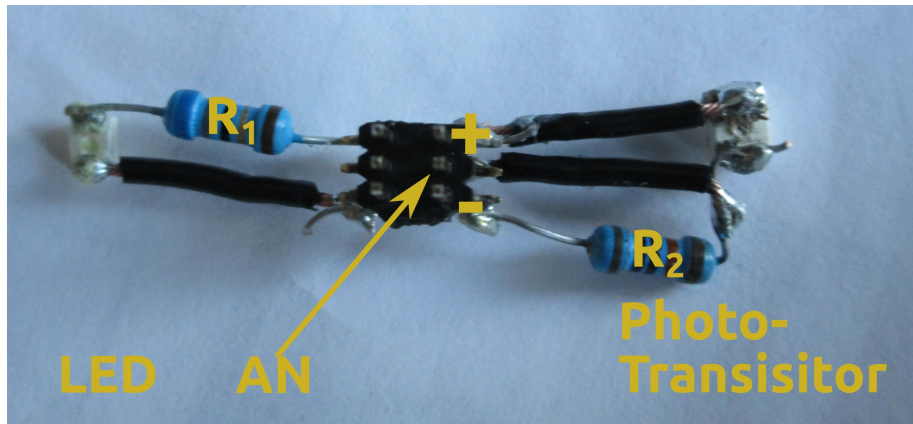


Figure 3: Monitoring device, top view

General Instructions

We will use the leads of the resistors, and some wire, as a skeleton for our device. Therefore, we will wait until the end to cut them. Soldering the LED and the PT can reveal challenging as both parts are very small. So, it is recommended to hold them firmly, for instance, using “helping hands”. It is easy to forget that both **LED** and **PT** should **face downward** as the light will be transmitted through the shell.

The LED

1. Before starting, ensure you can tell cathode and anode apart. Most “surface mounted LEDs” will have an indication such as a small chip on the + side.
2. Solder directly one lead of the $68\ \Omega$ resistor to the anode of the LED.
3. Cut a short wire (1 cm). Strip and twist its tip.
4. Solder one extremity of the wire to the cathode.

The Phototransistor (PT)

1. As for the LED, ensure you know the polarity.
2. Cut two short wires (1 cm). Strip and twist their tips.
3. Solder one extremity of a wire to the anode of the PT.
4. Solder directly one lead of the $100\ \text{k}\Omega$ resistor to the cathode. Leave some space between the anode and the resistor to solder the second wire.
5. Solder the tip of the second wire onto the lead of the resistor; between the resistor and the PT.

Assembling on the PCB header

1. Cut the PCB headed to leave only three columns (i.e. 6pins). From a bottom-up view, let us label the pins:

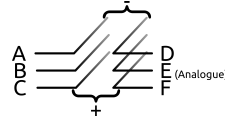


Figure 4: PCB header map

2. Solder the wire connected to the LED light on A. That is on the - side.
3. Solder the lead of the resistor connected to the LED on C **and** F (i.e. + side).
4. Solder the wire connected the anode of the PT to F.
5. Solder the leg of the resistor connected to the PT on A **and** D.
6. Solder the second wire of the PT on E.

Lead

1. Strip the tips of three long (>30cm) flexible and thin wires.
2. Cut three columns of the PCB socket.
3. Solder each wire to a different pin of the PCB header.
4. Plug the PCB socket on the pins D, E and F of the PCB header.
5. Unsure you can tell which wire is for ground (pin D), and which is for + (pin F).

Arduino

1. Plug the + wire on digital pin 2 – or 5 V if you want constant light (see ‘Fitting the Circuit’ section below).
2. Plug the ground wire on ground.
3. Plug the middle wire on analogue in 1.
4. Compile and upload the code for this project to the Arduino.
5. The LED light should start blinking quickly.

Fitting the Circuit

This is probably the hardest part of the process. For this reason, a four minutes video demonstrating this part of the protocol step by step is available online (see ‘Supporting Material’ section above).

The main steps are:

1. Protect the animal by putting tissue paper on the opening of the shell.
2. Plug the Arduino to the USB port of your computer.
3. Run the **python** program (see ‘Code Download’ section above). A window should appear and plot the signal in real time (see section ‘Python Code Complementary’ below).
4. Plug the circuit on. For now, *use the 5 V pin* instead of the digital pin as it will allow you to see the heart beating much better.
5. At this stage, you should be able to test the circuit by moving objects between PT and LED, and observe variation in the data displayed by the **python** program.
6. Move the LED on the shell until you see the beating heart with your eyes.
7. Use blu tack to maintain the circuit on the shell, and assess the quality of the signal displayed on your screen.
8. Move the PT until the signal to noise ratio is satisfying (i.e. sharp oscillations of large amplitude). Then glue PT and LED to the shell.
9. Unplug the circuit, interrupt the **python** program, and wait for the glue to dry.
10. Plug the circuit. This time, **use the analogue pin** as opposed to the 5 V.
11. Run the **python** program. In addition to simply display heart rate, the data can be saved to a text file using the `--out` option (see section ‘Python Code Complementary’ below).

In the end, the animal should have no difficulty to move with the device:



Figure 5: Snail with device

Important Note:

- Use as little glue as possible and **do not** touch the foot of the animal with any glue. If you are concerns about the toxicity of the glue, you may consider alternatives such as dental cement.

Arduino Code Complementary

This project only contains a short file. We start by defining numerical constants:

```
#define OVER_SAMPLING 16
#define FS 5.0 // in Hz
#define RISE_TIME 1 // in ms
#define POWER_PIN 2
#define PHOTO_TRANSISTOR_PIN 1
```

- Every data point will be the sum of `OVER_SAMPLING` consecutive reads.
- The sampling frequency `FS` (in Hz). That is the actual number of point we output every second.
- `RISE_TIME` defines how long should we leave the circuit turned on before reading any value.
- The digital pin controlling power is number 2.
- The analogue pin from which PT values are read is number 1.

Then, we compute the time to wait between consecutive reads:

```
const float time_to_sleep_ms = 1e3/(FS*OVER_SAMPLING) - RISE_TIME;
```

Since are performing *oversampling* every data point we output is, in fact, the sum of several (in this example, exactly 16) reads. Therefore, the actual reading sampling frequency $F_{s_a} = F_s * s$ where F_s is the resulting sampling frequency (i.e. 5 Hz) and s , the oversampling factor (i.e. 16 times). The delay dt between two actual reads is then simply $1/F_{s_a}$ (in seconds).

Note that we also subtract the rise time to be more accurate. Also, we multiply by 1000 since we want milliseconds instead of seconds.

In the setup:

```
void setup(void) {
    Serial.begin(57600);
    pinMode(POWER_PIN, OUTPUT);
}
```

We simply:

- Set the baud rate to 57600, which should be more than enough.
- Set the digital pin as an output pin that can take HIGH(5 V) or LOW(0 V) values.

And in the main loop:

```
void loop(void) {
    unsigned int accum = 0;
    for(int i =0; i < OVER_SAMPLING; i++){
        digitalWrite(POWER_PIN, HIGH);
        delay(RISE_TIME);
        accum +=  analogRead(PHOTO_TRANSISTOR_PIN);
        digitalWrite(POWER_PIN, LOW);
        delay(time_to_sleep_ms) ;
    }
    Serial.println(accum);
}
```

For each iteration:

1. We initialise an accumulator variable to 0.
2. We are going to sample as many time as the value of `OVER_SAMPLING`, so we start a for loop in which we:
 1. Turn the circuit ON.
 2. Wait for the circuit to be in a stationary state.
 3. Read the PT value.
 4. We increment the accumulator by the obtain value.
 5. We immediately turn the circuit OFF.
 6. We sleep for the delay `dt`.
3. When the above loop completes, we can send the value of the accumulator to the serial port in order to use it from a connected computer.

Python Code Complementary

This project uses `python` programming language for real time visualisation on a computer screen. Importantly, `python2` (not `python3`) should be run.

1. Install `python2` on your machine.
2. Install dependency packages using. You can use `pip`, the python package manager for that:

```
pip install numpy pyserial
```

3. Install `pygame` (<http://pygame.org/download.shtml>). **@edwbaker -> I am not sure how/if you want to display urls inline**
4. Download the `python` code (see ‘Code Download’ section above)
5. Open a terminal and run the `python` code. For instance:

```
python serial_monitor.py --port /dev/ttyACM0
```

Note that you may need to set a different serial port (e.g. `--port /MY/PORT/IS/HERE`) according to your machine and operating system. In addition, other options can be displayed by:

```
python serial_monitor.py --help
```

For instance, `--out /WHERE/TO/SAVE/THE/data.txt` can be useful.

When running the program, a visualisation window should appear:

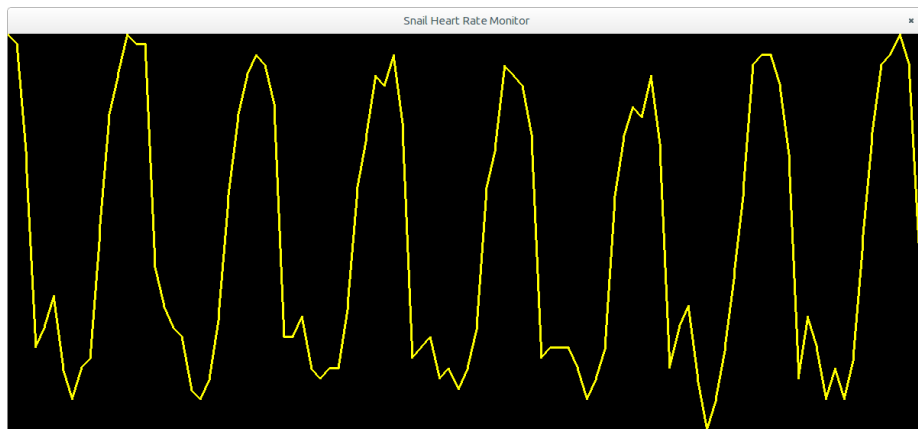


Figure 6: Real time visualisation window

If you are interested in understanding or adapting this `python` script, explanations are provided, as comments, within the source code.

Example of Result

This is an example of approximately one minute of real data, sampled at 10 Hz:

The corresponding power spectrum indicates a fundamental frequency around 0.75 Hz (vertical dashed line), which is 45 beats per minute:

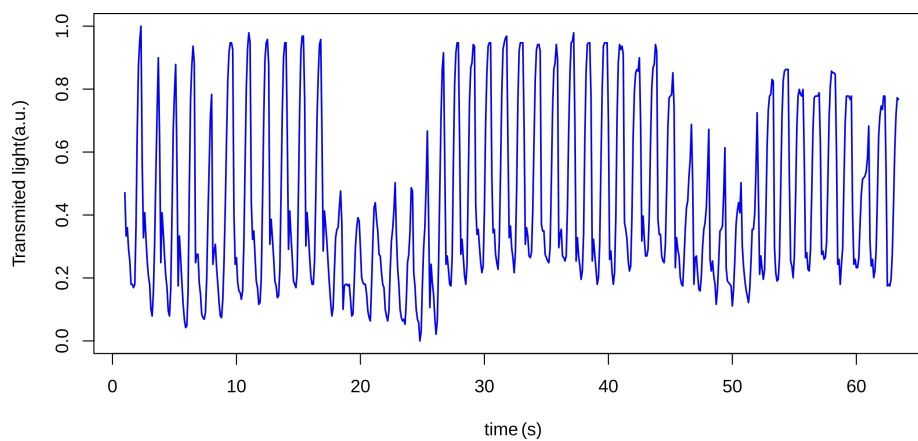


Figure 7: One minute of data

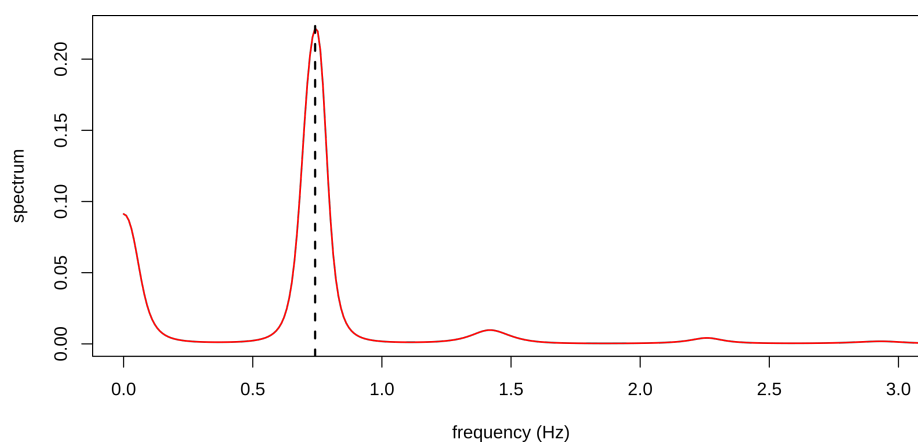


Figure 8: Power spectrum

Next steps

Improvements to this method are currently under investigation and include:

- Using infrared instead of red light in order to reduce environmental noise.
- Computing heart rate directly on the Arduino board.
- Developing a wireless version.
- Using several PTs to increase accuracy and robustness.