

ECG Display With Arduino



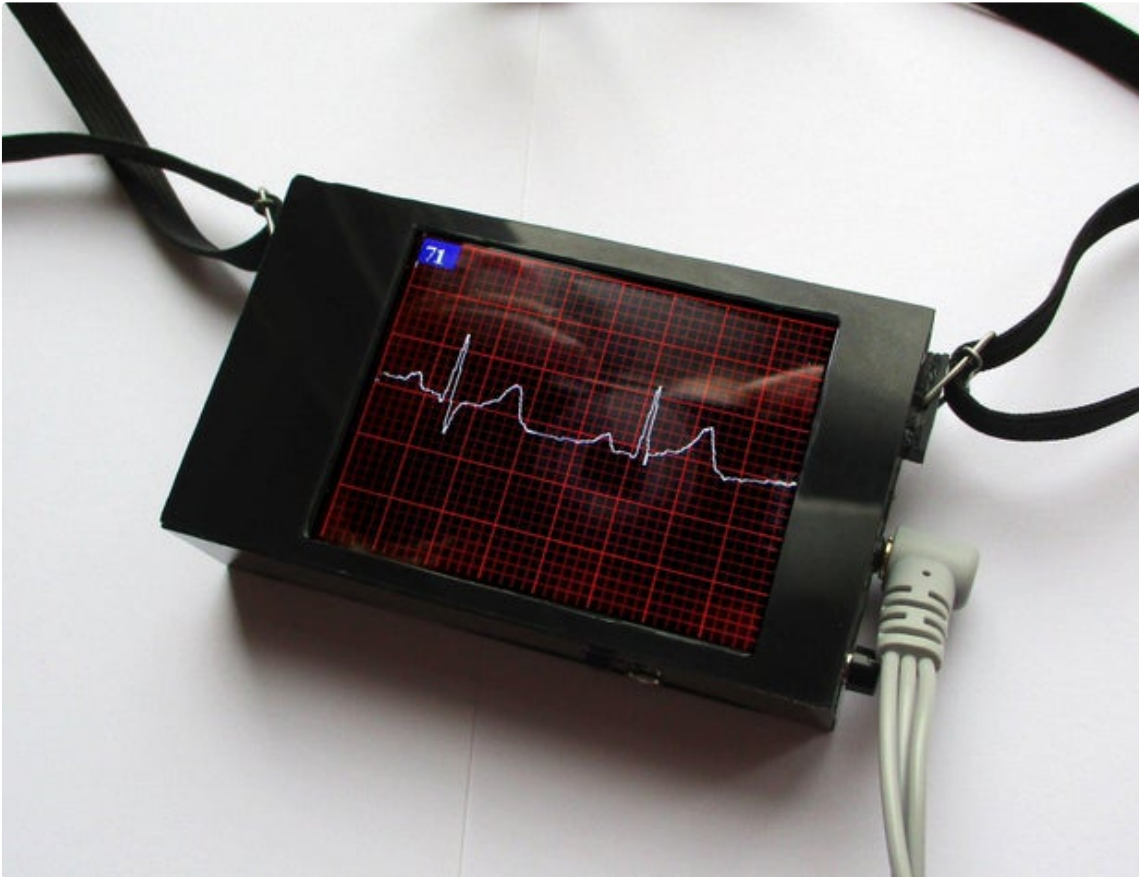
by Peter Balch

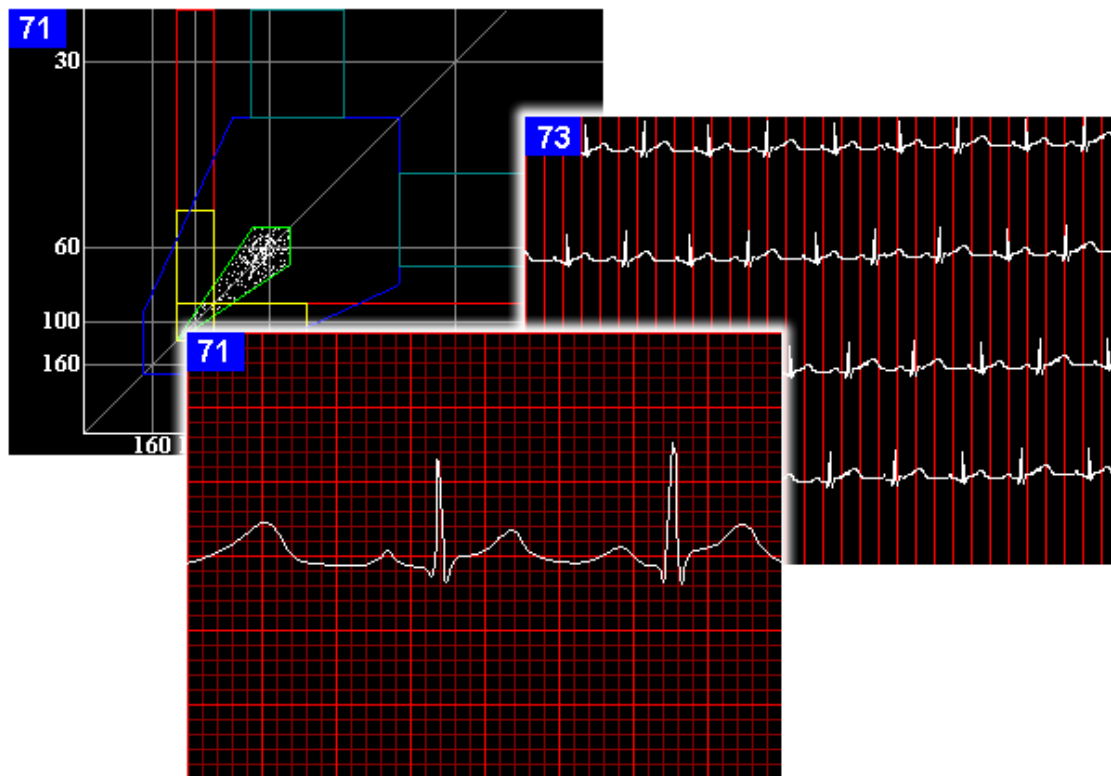
This ECG (Electrocardiogram) unit has an integrated display. The display can show individual heart beats in a large format, the train of heart beats in a small format and a Poincaré plot.

It is battery powered and can be worn round your neck on a lanyard.

It is built from an AD8232 module (£7 with leads), an Arduino Nano (£3) and a 2.8" TFT display (£6) plus a few resistors, a pushbutton and so on - the sort of things you already have. For power, you can use 4 AA cells or a single Lithium cell.

You might be interested in your heart during the Covid-19 pandemic. Covid-19 can sometimes give rise to cardiac complications including arrhythmia, ischaemia, infarction, tachycardia and conduction problems. But by the time that happens, you're probably lying in an intensive care ward.





Step 1: Safety and Medical Disclaimer

Here's some legal blurb and a discussion of electrical safety. The most important point is **do not operate the circuit when it's plugged into a PC which is plugged into the mains**. Oh yes - and don't blame me if you kill yourself.

I am not a medical professional. This Instructable was created for information purposes only. It does not describe a medical device. The information given here is not intended to be a substitute for professional medical advice. If you think you have a medical condition you should contact a qualified health professional. If you think you have a medical emergency you should contact your local emergency services. The device described here is not a substitute for professional medical diagnosis, ECG interpretation or treatment. You are encouraged to confirm any information in this Instructable.

As I have no control over how you build the electronics described here, I cannot guarantee its safety as a medical device. I think it is perfectly safe for me to use but you should make up your own mind. It's running off 4 AA cells or a single Lithium cell for goodness sake - how dangerous can it be?

If you have been fitted with an internal defibrillator or pacemaker, do not attach electrodes to your body and mess about with circuits.

Professional medical electronics should conform to [IEC 60601](#). I have designed medical electronics in the past but it was a few years ago. I seem to have lost my copy of IEC60601 which is annoying as you have to pay for it. Would this device get IEC 60601 approval? How safe is it?

IEC 60601 is a huge document. The most interesting questions to an electronics designer are:

- how much current can I pass through the patient?
- what electrical isolation is required?
- what happens if a "component" fails?
- how much electromagnetic interference can I produce?
- what about a defibrillator?

How much current can I pass through the patient? In "Normal Working" operation the current limit is 100uA a.c. and 10uA d.c. and in single fault condition it is 500uA a.c. and 50uA d.c. This circuit would be classed as an "Applied Part".

I don't know what circuit the manufacturer of the AD8232 module has used. When I examine the one I have, it appears to be one of the circuits in the "Application Circuits" section of the AD8232 datasheet.

The input impedance of the chip pins is 10Gohm so the "Normal Working" operation the current is under 1nA.

The electrodes are connected to the chip through 180k resistors. The chip operates at 3.3V so if the chip goes wrong, the maximum current is $3.3/180k = 18\mu A$.

So the chip and its circuit is safe so long as it's battery operated.

What electrical isolation is required? An "Applied Part" has to be double isolated at 4000VAC. Your desktop PC is not electrically isolated to that degree therefore you should never operate the circuit when it's plugged into a PC which is plugged into the mains. A standard PC power supply almost certainly does not comply with the electrical IEC 60601-1 requirements from several standpoints, e.g. leakage current, dielectric strength.

Can you operate it when it's plugged into a PC which is running on batteries? It wouldn't be allowed for a medical device but for a domestic device it probably would be allowed. (Domestic regulations are less stringent.)

Did I operate it when plugged into a PC which was plugged into the mains? I'm not saying. I've been using desktop PCs for decades and have never got a shock off one. I'm just telling you that you shouldn't do it.

Would the circuit's isolation conform to IEC 60601? If it were battery operated then that's one layer of isolation. Where's the second layer? I guess maybe you'd need two insulated boxes one inside another. IEC 60601 sometimes seems a little silly.

The circuit should have no exposed conductive surfaces that are connected to the electronics. You have to worry about electrostatic discharge: what if the patient rubs a balloon on their hair then touches the box?

The patient shouldn't be able to touch the electronics or the battery or whatever. Yes, I know you've never got a shock off a 6V battery - I'm just telling you the regulations.

Electrical isolation also depends on the creepage distances on the PCB and air clearances. The AD8232 module uses 0603 resistors on its input. An 0603 doesn't have sufficient creepage.

Is the electrode connector suitable? No way. It's a 3.5mm stereo jack plug. You could tape the electrodes to your chest then poke the jack plug into a mains outlet. I don't know why you'd want to but you could - right?

What happens if a "component" fails? IEC 60601 says that the device must remain safe under a 'single fault condition'. A medical device must operate safely not only in normal conditions, but also in abnormal and single fault conditions. The standard does not require that the medical device remains safe with two or more independent faults. A "fault" is typically a component failure.

What counts as a "component"? I have no idea. You could go through each of the electronic components and say "what happens if this goes short-circuit?" and "what happens if this goes open-circuit?". Is the PCB a "component"? How can it fail? Can it suddenly re-arrange itself and connect battery to two electrodes? I can't find a definition of a "component" or "fault".

If it's battery operated and a "component" is an electronic component like a resistor or an IC then the circuit is safe.

How much electromagnetic interference can I produce? EMC is a big problem when designing medical equipment. Hospitals are full of electronic stuff that people's lives depend on so the EMC requirements for medical equipment are a lot tougher than for domestic electronics.

I have no idea how much EMC the board produces. The biggest culprit is probably the Arduino. I've never had an Arduino interfere with a TV or cellphone but I suppose it might.

In theory, you should put the circuit in a Faraday cage and worry about what wires go in and out. In practice, I don't suppose you've ever thought about EMC when you build an Arduino circuit, have you?

What about a defibrillator? So you've built your medical electronics and done the safety analysis and then IEC 60601 has one more surprise for you. Your circuit has to survive a defibrillator.

What is the standard for a defibrillator? I couldn't find one. Each manufacturer decides for themselves and the physician can decide what to apply. Shall we guess a capacitor charged to 300V containing 300 joules? That's connected to your chest somewhere near the ECG electrodes. I suspect that will fry the circuit. IEC 60601 says that the device must remain safe - it doesn't say that it has to continue to work.

A proper medical ECG can survive a defibrillator and go on working. Normally, you would have some protection diodes on the connections to the electrodes. The AD8232 module doesn't have them.

The bottom line is: you could make this circuit into a medical device that would conform to IEC 60601 but it would be a lot of work and testing and certification would be a huge expense. This is just a circuit you're building for fun.

See [here](#) and [here](#) for discussions of the dangers of a small battery. I have measured the resistance between the electrodes as 150k so, in the worst case, a 6V battery gives a 40uA current - well below the milliamps needed for a battery to be dangerous.



Step 2: AD8232 Module

The [AD8232 chip](#) contains a high quality, low noise instrumentation amplifier and signal conditioning to remove noise. It is intended primarily for recording ECG and takes a lot of the hard work out of designing a system.

Buy a module like the one in the photo. Search eBay for "AD8232 module" or "ecg module". For convenience, get one with leads and electrode pads.

Connect it to an Arduino Nano (or Uno or Mini with a 16MHz 328P) as shown above. I used a solderless breadboard.

The AD8232 includes "leads off detection". If either lead is disconnected, the LO+ or LO- pins of the module go high. For initial testing, I connected the LO+ and LO- pins to two LEDs though 1k resistors to ground. The LEDs light when a lead is disconnected. For the final circuit, remove the LEDs and connect LO+ and LO- to the Arduino. Don't leave the LEDs connected - the AD8232 pins won't go high enough for the Arduino inputs to register HIGH.

Download the [ArdECG0.ino](#) sketch and upload it to the Nano. Connect the Nano to a PC running on batteries. In the Arduino IDE, select the Tools|SerialPlotter menu command. Set the baud to 57600.

With the module I bought, the electrode leads are coloured:

- LA Left Arm: Green
- RA Right Arm: Red
- RL Right Leg: Yellow

Yours may be coloured differently. Which lead is which? try touching the leads together:

- Non touching: both LEDs lit
- LA touching RA: both LEDs lit
- LA touching RL: LO+ LED is off
- RA touching RL: RO- LED is off

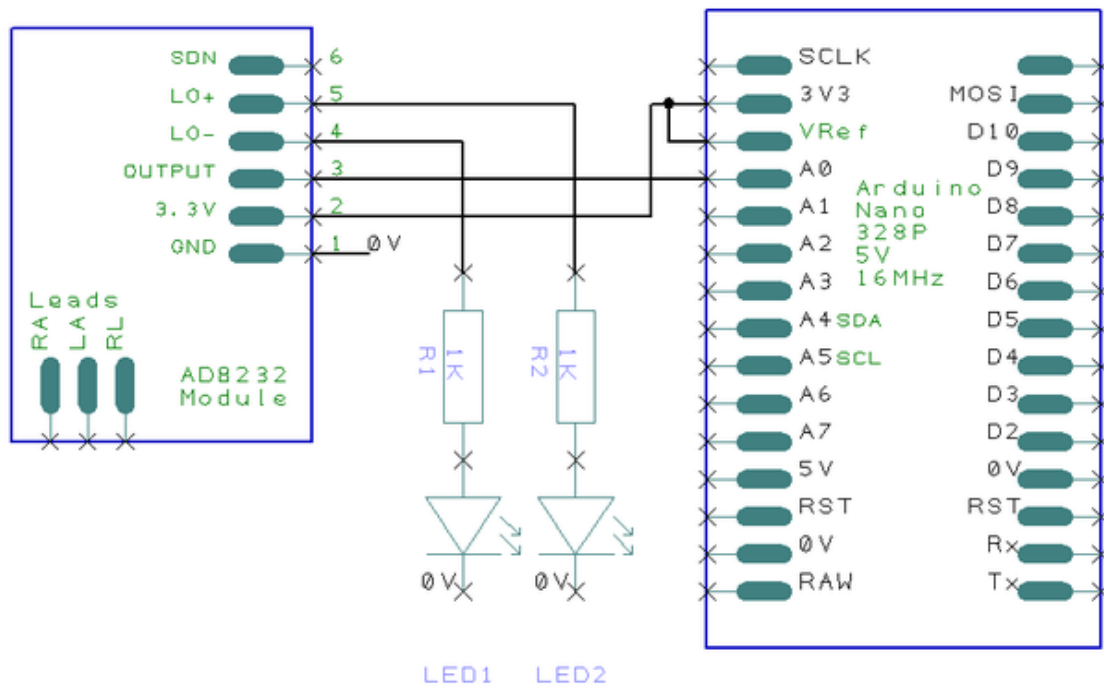
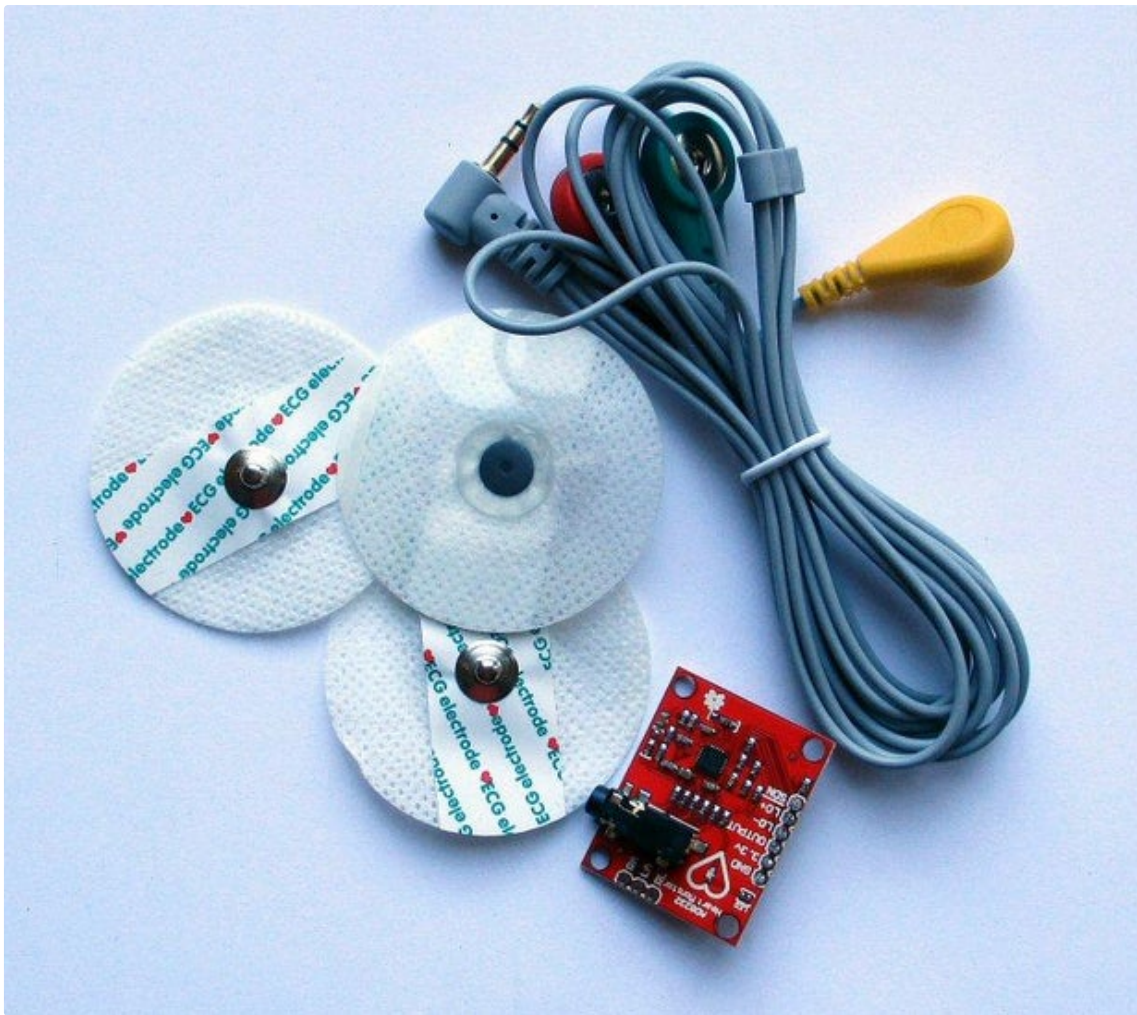
My module came with only 3 sticky electrodes. They'll get used up quickly so I decided to make my own. I happened to have some of the popper connectors left over from another project. I used tinplate for the electrodes themselves. Copper coins work as well but tarnish more quickly and stain your skin. As electrode jelly, I mixed thick shampoo and salt. I stuck them onto my chest with masking tape - it would last a few hours.

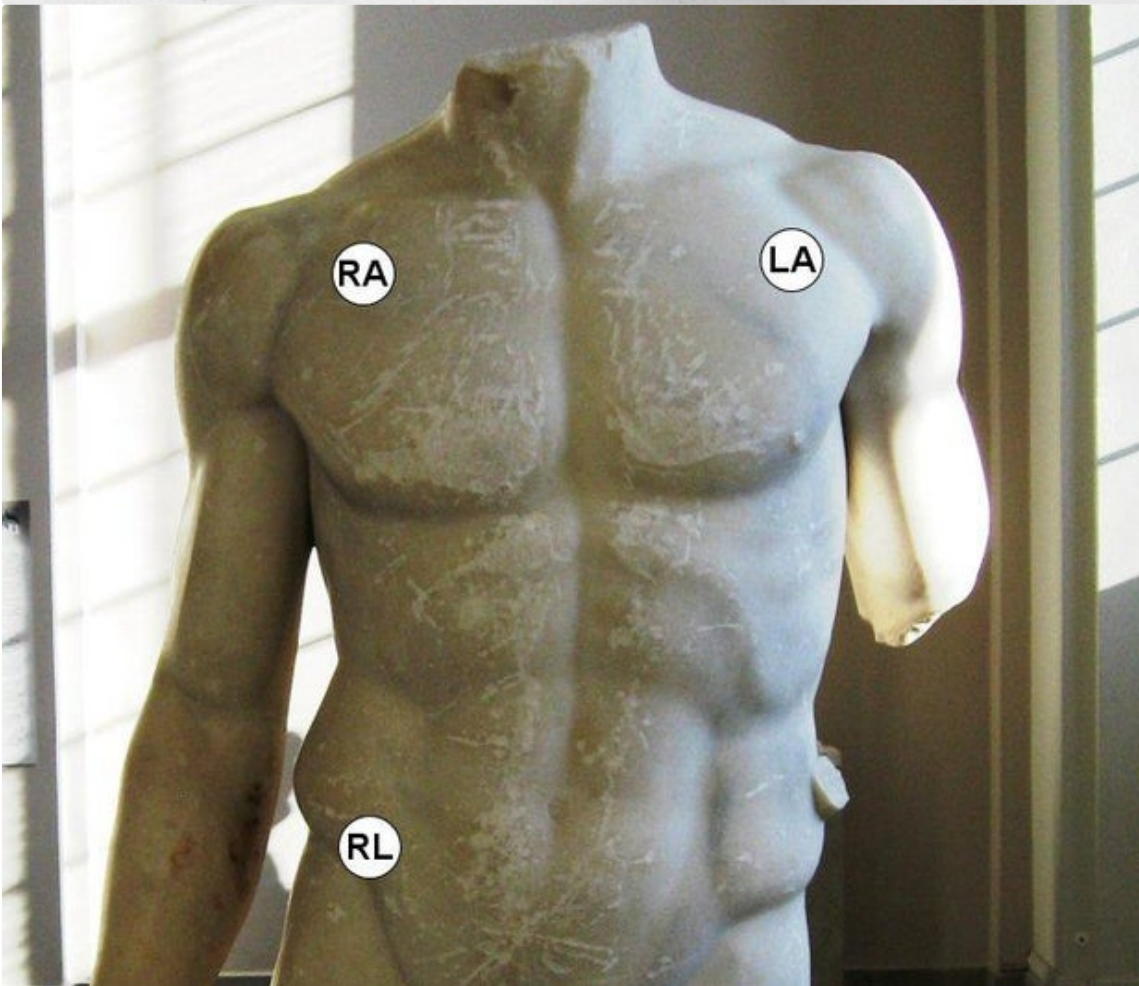
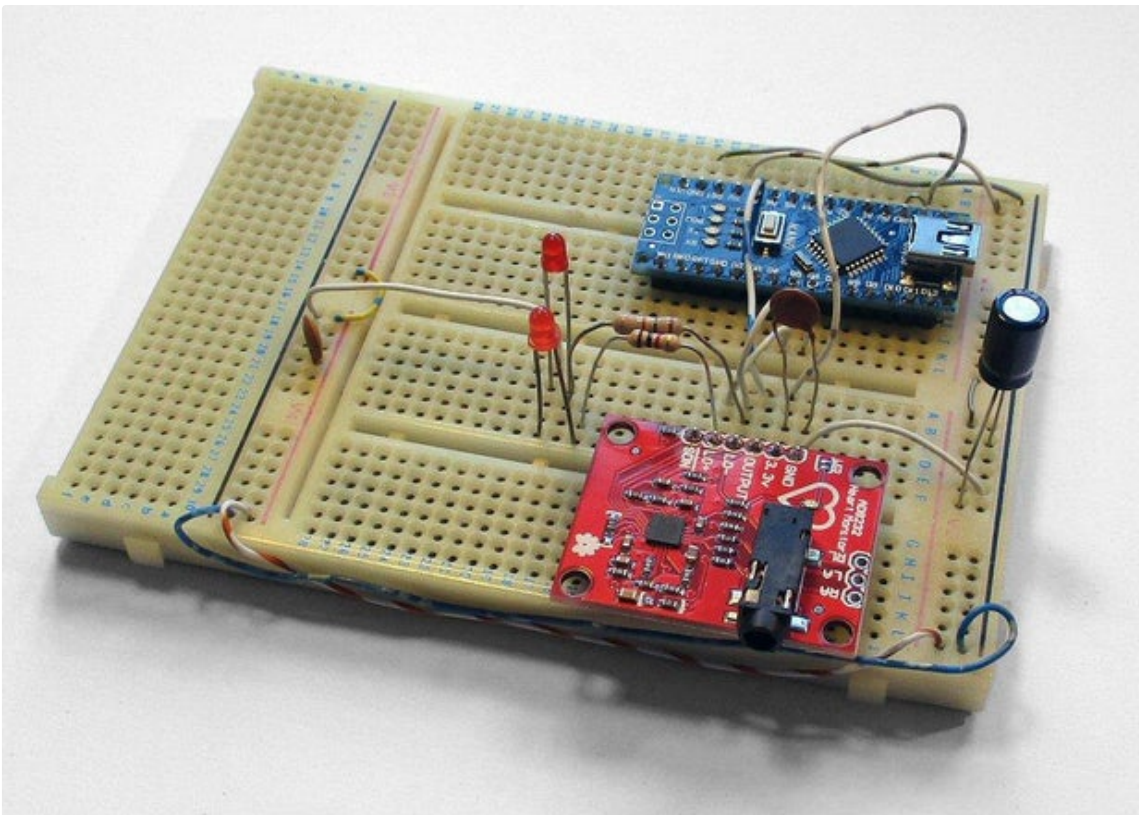
Attach the electrodes as shown in the picture above.

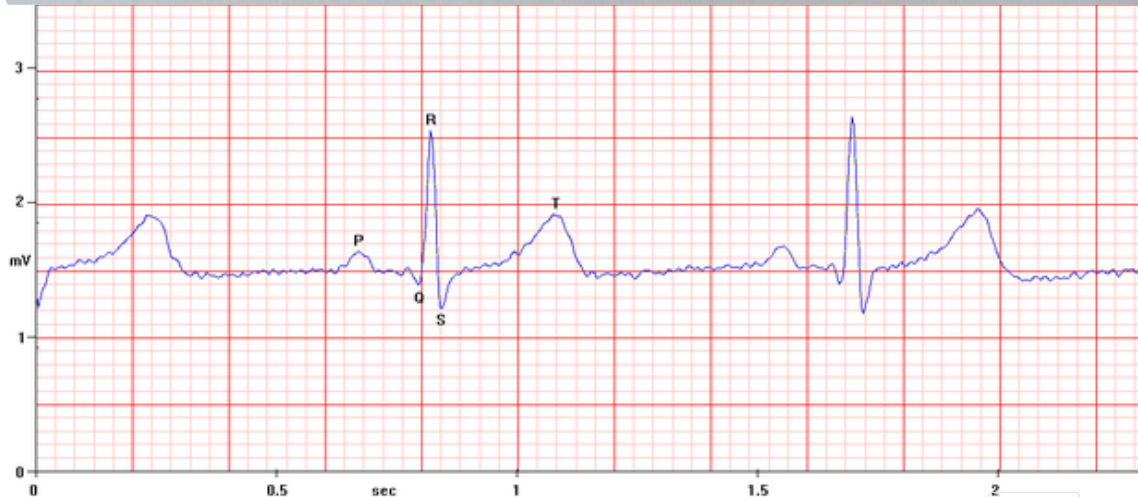
- LA to the left side of your chest below the clavicle (collar bone)
- RA to the right side of your chest below the clavicle
- RL somewhere low down away from the heart

The RL electrode is called "Right Leg" but it doesn't actually need to be on your leg. It's a [reference electrode](#) that reduces common-mode interference. Anywhere well below and away from your heart is fine. The lower-left of the abdomen is good. Try to avoid muscles - their movement and electrical activity could interfere with the signal.

Google for "ecg 3 lead electrode placement" for other diagrams.







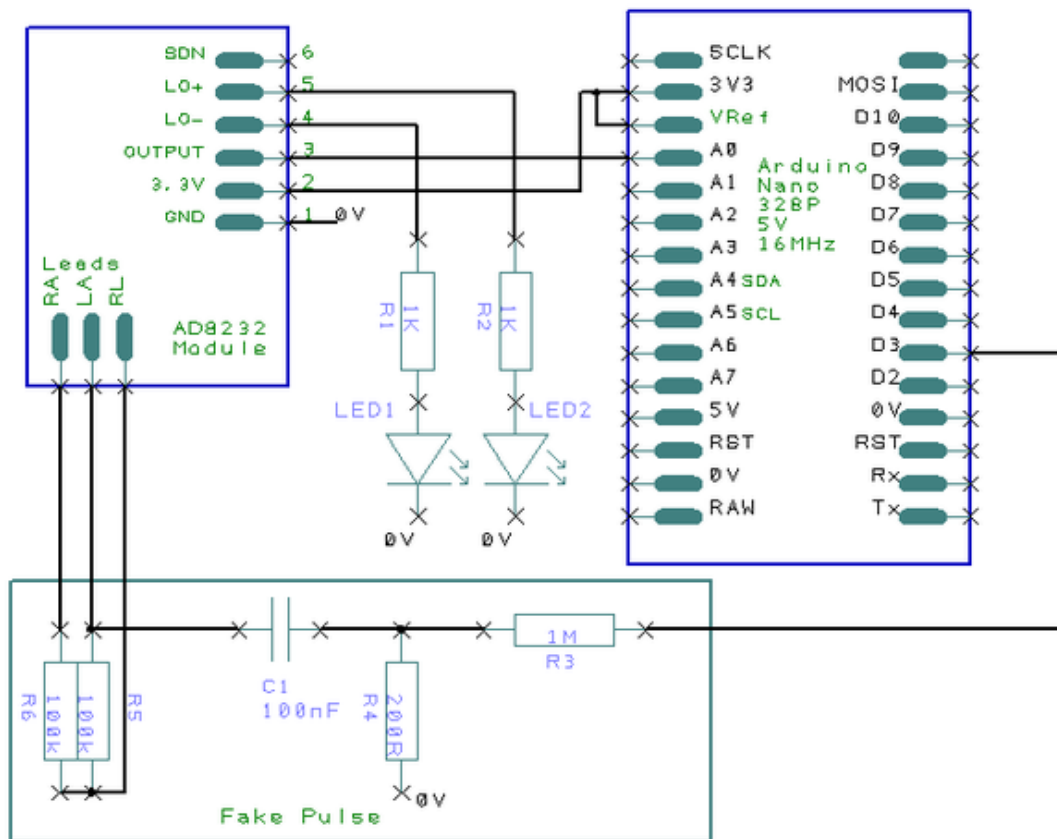
<https://www.instructables.com/ORIG/F3W/8230/KNOLY45J/F3W8230KNOLY45J.ino>

Download

Step 3: Fake Pulse

The ArdECG0.ino sketch can produce a fake "pulse". It's not a very realistic fake but it produces pulses of about the right size at about the right frequency. It's convenient to use it when debugging your unit.

The 100k resistors persuade the AD8232 that there are electrodes attached. There are some pads on the AD8232 module PCB labelled RA, LA, RL. I just poked some wires into them.



Step 4: The Heart

Doctors think of the heart as a complex organ but to an electronics designer, it is simply a generator of electrical potentials - several tens of millivolts.

The sinoatrial node (SA node) acts as an oscillator generating a pulse every second or so. Electronically, it's a relaxation oscillator (or Van der Pol oscillator).

The pulse travels slowly to the muscles of the atria and to the atrioventricular node (AV node). The muscles of the atria contract and then relax. As they contract, they pump blood into the ventricles. As they relax, the atria take in blood through the veins from the body or the lungs.

The AV node delays the pulse (by 120-200mS) then sends it to the muscles of the ventricles. Electronically, it too is a relaxation oscillator but its period is much longer than the SA oscillator. It is "prematurely" triggered by the SA oscillator so they become entrained. If the SA oscillator fails, the AV oscillator can oscillate on its own as a backup system.

The muscles of the ventricles contract and then relax. As they contract, they pump blood to the body or the lungs.

The whole process takes around 500mS.

It takes more power to pump blood round the body so the biggest signal we see is from the muscles of the left ventricle.

Sodium, potassium and calcium ions are pumped across the membranes of the cells of the heart muscles. When a heart muscle is relaxed, the difference in ion concentrations causes a 90mV potential difference across the cell membrane. Inside the cell is more negative.

When an action potential occurs a muscle contraction is triggered, Sodium channels open, Na⁺ ions rush into the cell by diffusion and the voltage goes to zero. That's called "depolarisation". After a 100mS delay, potassium channels open, K⁺ ions rush out of the cell and re-establish the 90mV resting potential. That's called "repolarisation". Ion pumps then restore the ion concentrations. (Calcium ions are also involved but I'm ignoring them.)

When an action potential occurs in a cell, the cell next to it is triggered. So the action potential spreads through the muscle and also via the Purkinje fibers. It's a slow process; no fast nerve conduction involved.

- Muscle cell conduction is 0.3–0.4 m/s.

- Purkinje fiber conduction is 2–3 m/s.
- Normal neuron conduction is 70-120 m/s.

Purkinje fibers can oscillate by themselves at 20-40 bpm and so act as a backup system if the SA and AV oscillators fail.

An ECG records the action potentials of the different muscles. The action potentials of the muscles are transmitted through the chest wall and skin by simple electrical conduction. So the conduction is fast. By the time it reaches the skin the signal is around 1mV.

Because different action potentials occur in different regions of the heart, by repositioning the ECG electrodes you can record activities in different muscles. With the electrodes at the left and right under the clavicles, you'll get the classical PQRS shape you see in a textbook (called "Lead I"). If you position the electrodes elsewhere you can pick-up other details of heart activity; different positions for pairs of electrodes are called "axes".

It's confusing, when doctors talk about "leads" and "axes". When they say "lead", they don't mean a lead - i.e. an electrode. Remember, what an ECG displays is the difference between two electrodes - it's a differential amplifier. The LL or RL electrode acts as a ground to help with common-mode rejection. A "12 Lead" ECG has 6 electrodes called V1 to V6 which are in a line in front of and to the bottom-left of the heart. Then there are RA and LA near your armpits. There may be two reference electrodes: LL and RL; the extra one doesn't seem to do much so there's usually just one. Pairs of electrodes are also called "leads" - LA-RA is called "Lead I", LL-RA is called "Lead II" and LL-LA is called "Lead III". It's a mess.

The correct placement is:

- V1 is placed on the 4th intercostal space on the Right side of the sternum
- V2 is placed on the 4th intercostal space on the Left side of the sternum
- V4 is placed in the 5th intercostal space in a perpendicular line from the left nipple
- V3 is placed between V2 and V4
- V6 is in the 4th intercostal space in the left mid-axillary line (from the armpit to the hip)
- V5 is placed between V4 and V6

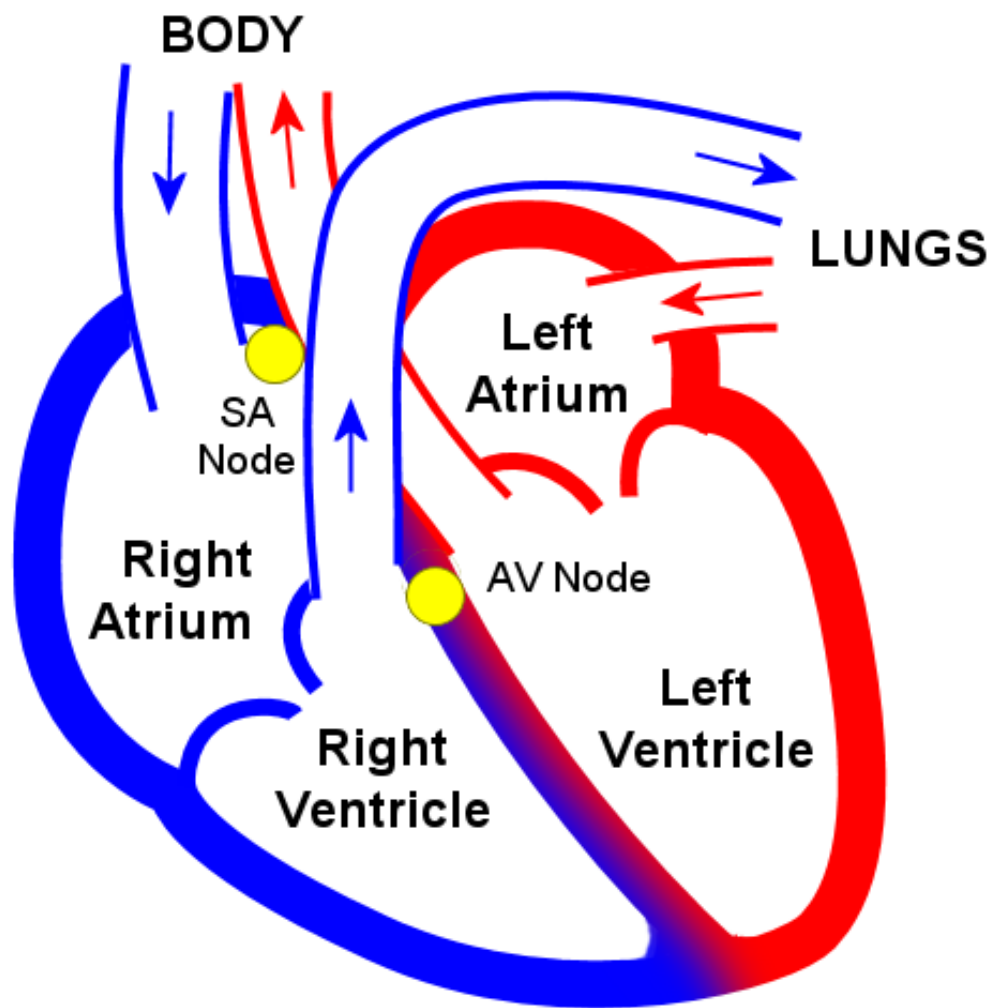
(Right and Left are patient Right and Left. On most males, the 4th intercostal space is between the nipples and is the indentation or soft area between the ribs.)

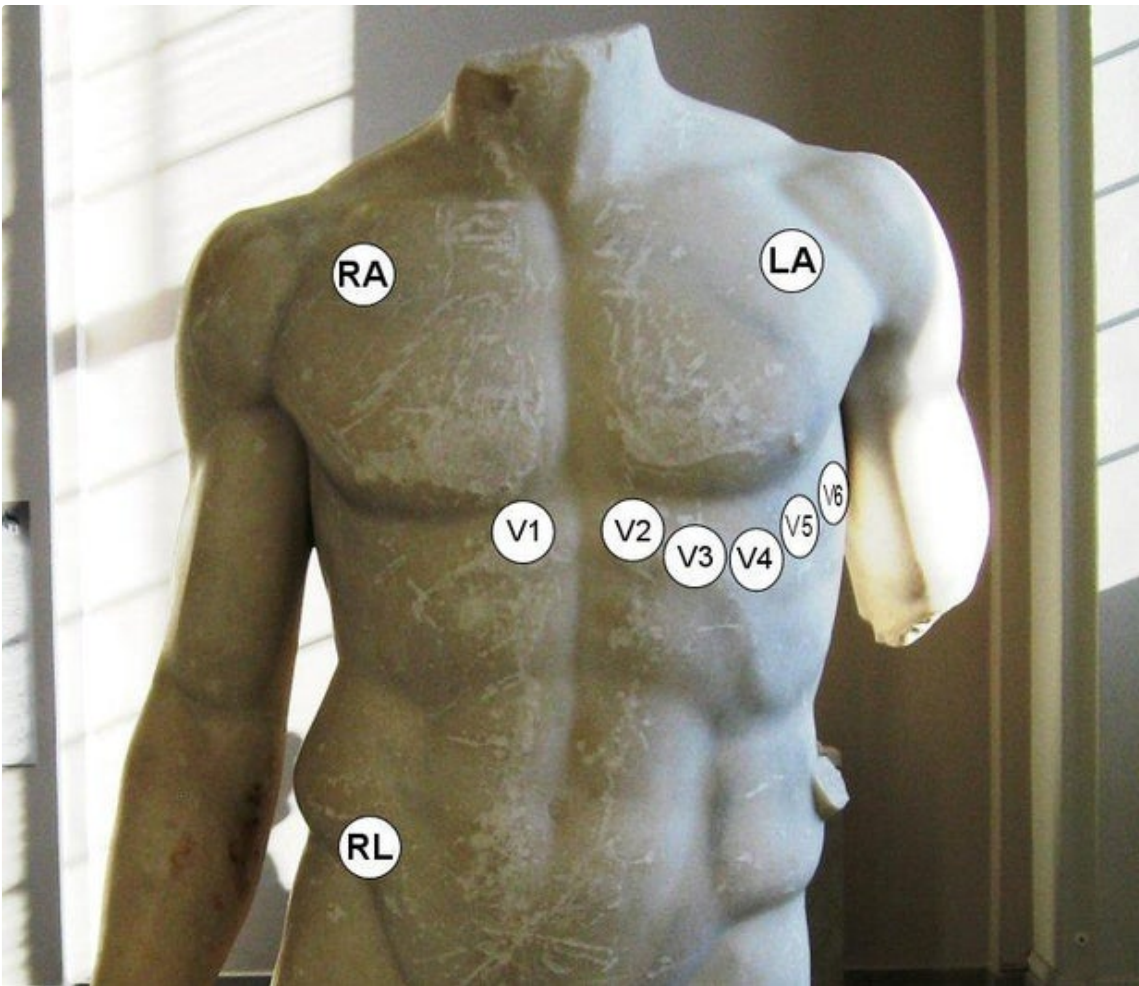
The maths is:

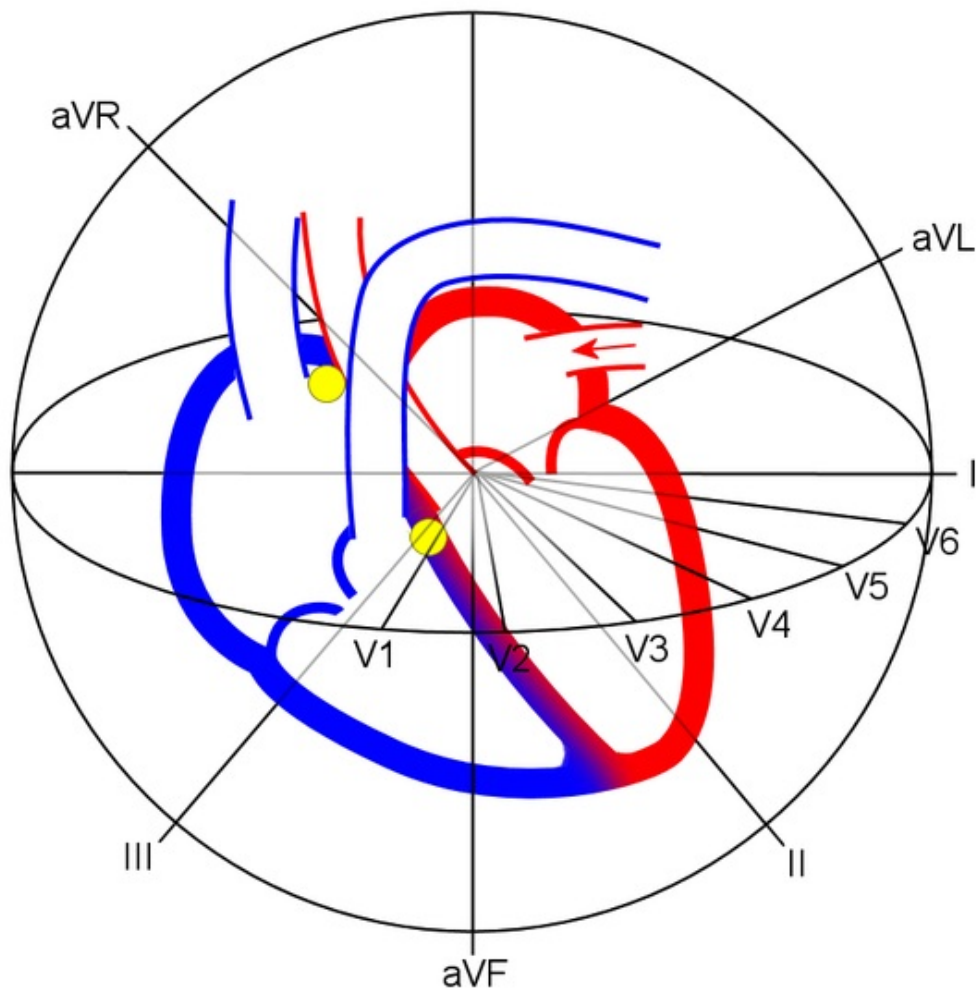
- Lead I: LA - RA
- Lead II: LL - RA
- Lead III: LL - LA
- Lead aVR: RA - average of (LA,LL)
- Lead aVL: RL - average of (RA,LL)
- Lead aVF: LL - average of (RA,RL)
- Lead V1: V1 - average of (RA,RL,LL)
- Lead V2: V2 - average of (RA,RL,LL)
- Lead V3: V3 - average of (RA,RL,LL)
- Lead V4: V4 - average of (RA,RL,LL)
- Lead V5: V5 - average of (RA,RL,LL)
- Lead V6: V6 - average of (RA,RL,LL)

The different "leads" are like looking towards the centre of the heart from different directions in 3D (see diagram above). The directions are called "axes". To add more confusion the heart also has an "axis" - the electrical activity is symmetrical about that axis. If part of the heart goes wrong, the heart's axis might have "deviation". If "leads" I, II and III look roughly the same, that's normal; if they're different, that's "deviation".

Broadly, what can go wrong with the whole system is that the nodes (oscillators) or the different muscles stop functioning properly. For instance, parts of the muscles might be starved of oxygen and they don't produce an action potential. Or the SA node might run at the wrong rate or the SA and AV nodes might become disconnected and the oscillators run at different rates.







Step 5: Adding a Display

The display is a 2.8" colour TFT LCD screen with a ILI9341 controller, 320x240 pixels. I chose a 320x240 SPI display because can be updated reasonably quickly and uses few Arduino pins.

Search eBay for "320 240 TFT SPI" and you'll find a variety of displays. Get one that looks the same as the photo. A display without a touch screen is cheaper.

I don't use the touch screen function in this project. I want to run the unit on 4 AA cells or a Li-ion cell. As the AA cells discharge their voltage falls. I've found that the Arduino, display and AD8232 work fine with a supply voltage as low as 3V but the touch-screen stops below 4.5V.

The LCD has the following pins:

- VCC 5V
- GND ground
- CS LCD chip select
- RESET reset
- DC data/command
- MOSI SPI bus MOSI of Arduino
- SCK SPI bus SCLK
- LED back light
- MISO SPI data out of LCD (ignored)
- T_CLK SPI bus SCLK (ignored)
- T_CS touch chip select (ignored)
- T_DIN MOSI of Arduino (ignored)
- T_DO SPI data out of touch (ignored)

- T_IRQ touch interrupt request (ignored)

The display I bought has a built-in 3V3 regulator. If you look on the back of the PCB, you'll see it connected to the VCC pin. It's a 662K chip. So the module can be powered by 5V and you can connect it directly to your 5V Arduino pin. The LED power pin can also be connected directly to the 5V pin.

(Check that your display has a 3V3 regulator. If it doesn't and it requires 3.3V power then it takes around 50mA which is just at the limit of what the Arduino Nano's 3.3V regulator can provide. So it may be a mistake to power the display from the Nano's 3.3V pin. You'll should supply your own regulator.)

The logic pins of the display require 3.3V signals. You can't connect them directly to the 5V I/O pins of the Arduino. I've used resistors to drop the voltage.

Adafruit very kindly publish an ILI9341 library and several other libraries are available in Github and elsewhere. I tried a few and didn't like any of them. Some simply didn't work and all were huge. You write an Arduino sketch that draws a line and some text and you find your memory is 75% full. So I wrote my own drivers.

Wire up the display as shown and download the ArdECG1.ino sketch.

I've decided I don't like the way the Arduino IDE uses "libraries". It makes version control difficult. So I now keep the code needed to compile a sketch together. Put all these files in the same folder:

- ArdECG1.ino
- SimpleILI9341.h
- SimpleILI9341.cpp

SimpleILI9341 has a standard set of drawing commands very similar to all such libraries.

Some of the "fast" libraries you can download use special timing loops and are upset when other, maybe slower, devices are used on the same bus. SimpleILI9341 is written in C rather than assembler so isn't quite as fast as it could be but is much more portable and it shares the SPI bus politely with other devices. A [Windows program](#) can be downloaded which allows you to make your own fonts and icons.

You can download the [ILI9341](#) data sheet from the web. You send it a command by

```
set DC low
set CS low
send a command byte
set CS high
```

and send it data by

```
set DC high
set CS low
send zero or more data bytes
set CS high
```

You can see how I do it in the tft_write functions. The data bytes might be a whole row of pixels or a setting for a control register.

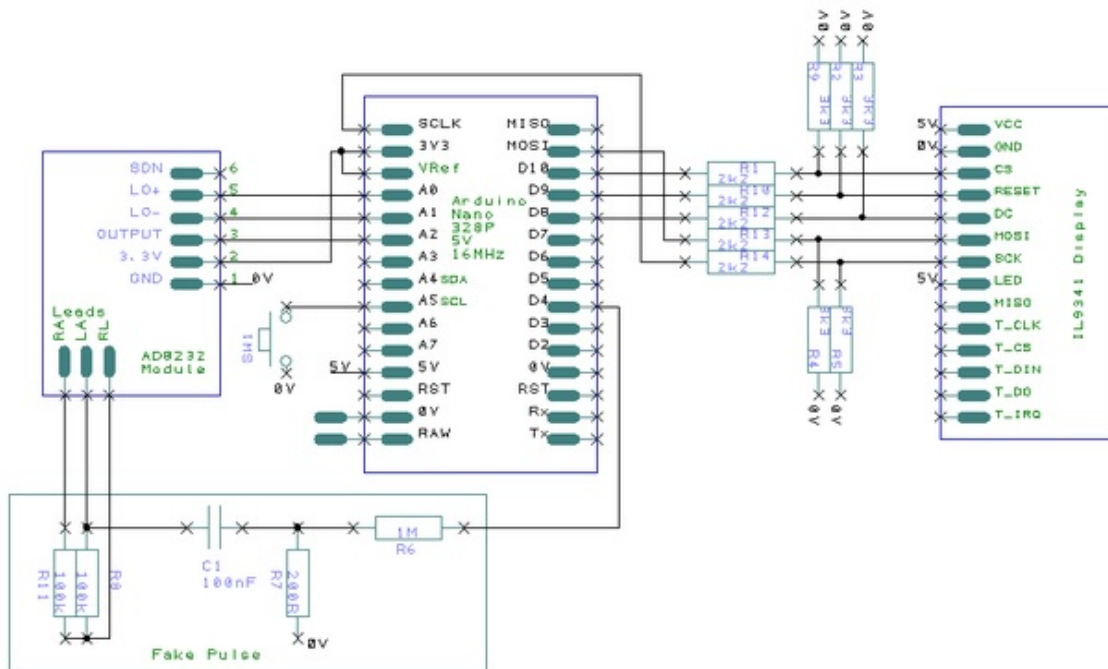
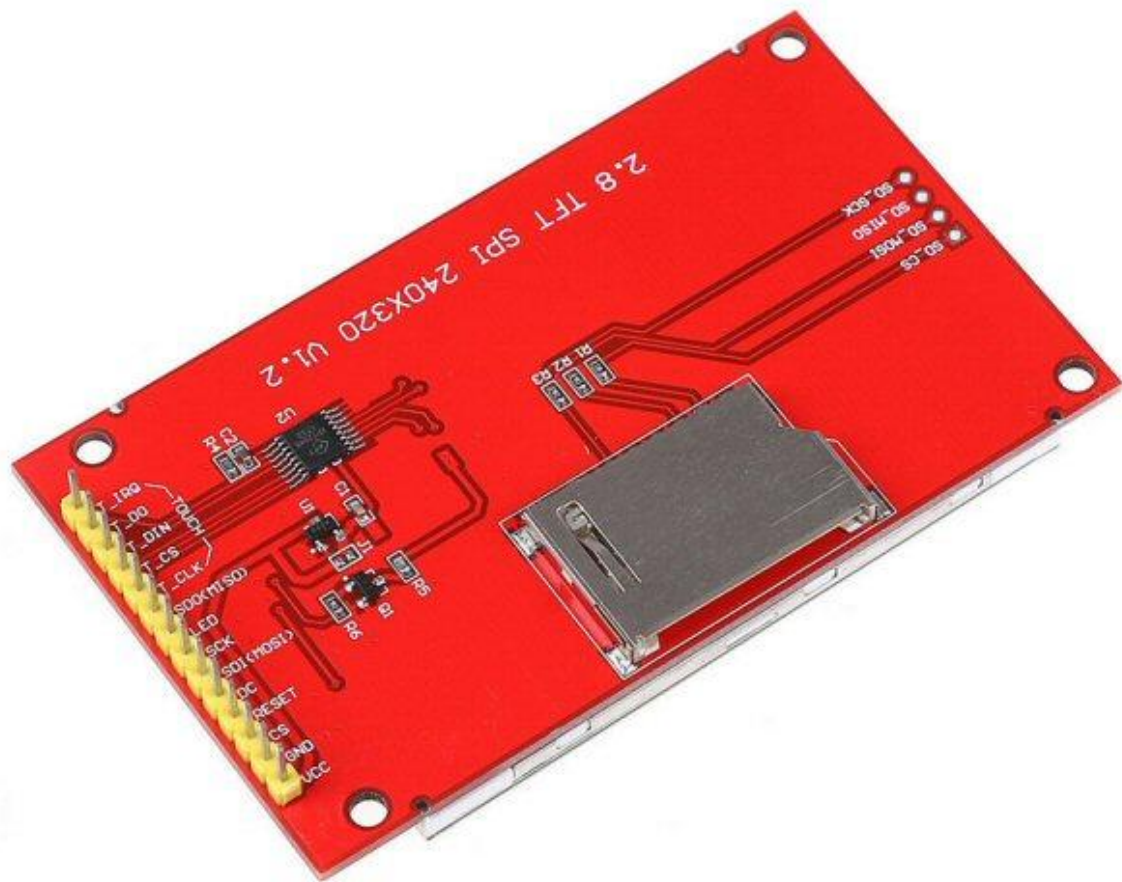
The ILI9341Begin() function in the library shows you the initialisation command set I've chosen. You might want to change the commands if you choose a different ILI9341 display (e.g. with more pixels) or want a different orientation. I hope my code is easy for you to see how to change if you need to.

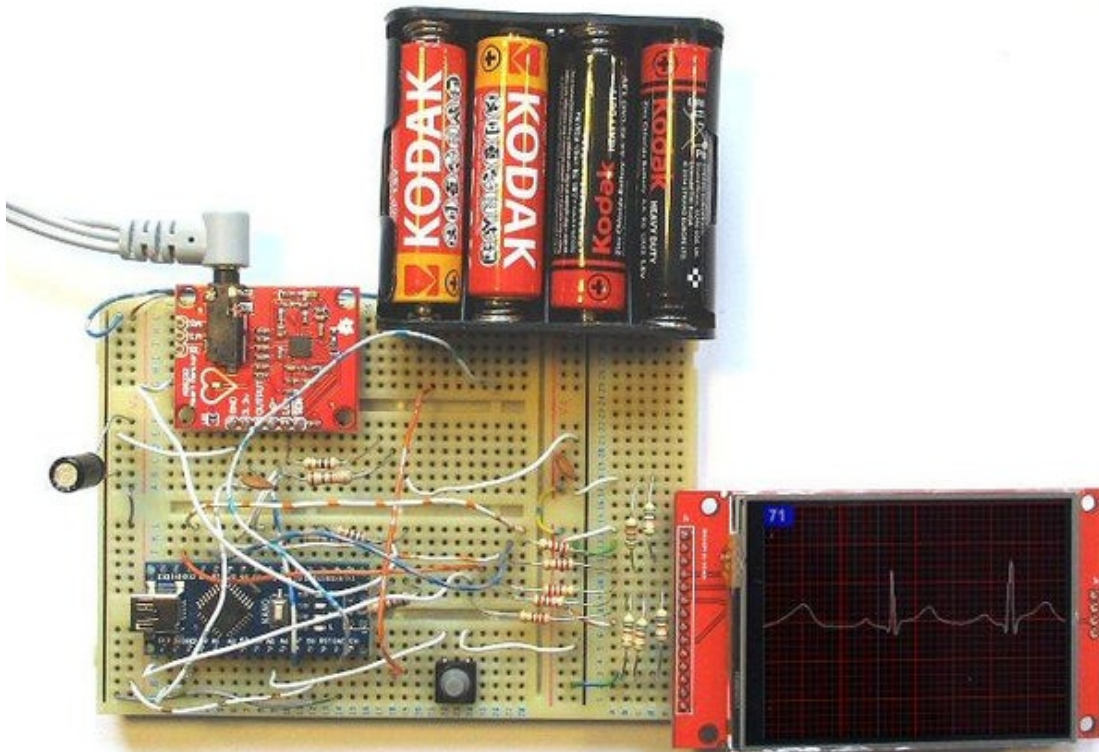
In the circuit diagram are components in a box labelled "Fake Pulse". Do not include them in your final build. They are useful during development and debugging. In the ArdECG1.ino sketch there is a #define to enable/disable fake pulse generation. It's somewhere near the top:




```
#define bHasFakeECG
```

Comment out the line to remove the fake pulse.







	https://www.instructables.com/ORIG/F90/0H8R/KMW14CJT/F900H8RKMW14CJT.ino	Download
	https://www.instructables.com/ORIG/FX7/1GB1/KMW14CJU/FX71GB1KMW14CJU.cpp	Download
	https://www.instructables.com/ORIG/FCR/KZHV/KMW14CJV/FCRKZHVKMW14CJV.h	Download

Step 6: Display Modes

I've used a pushbutton to cycle round the different views of the ECG:

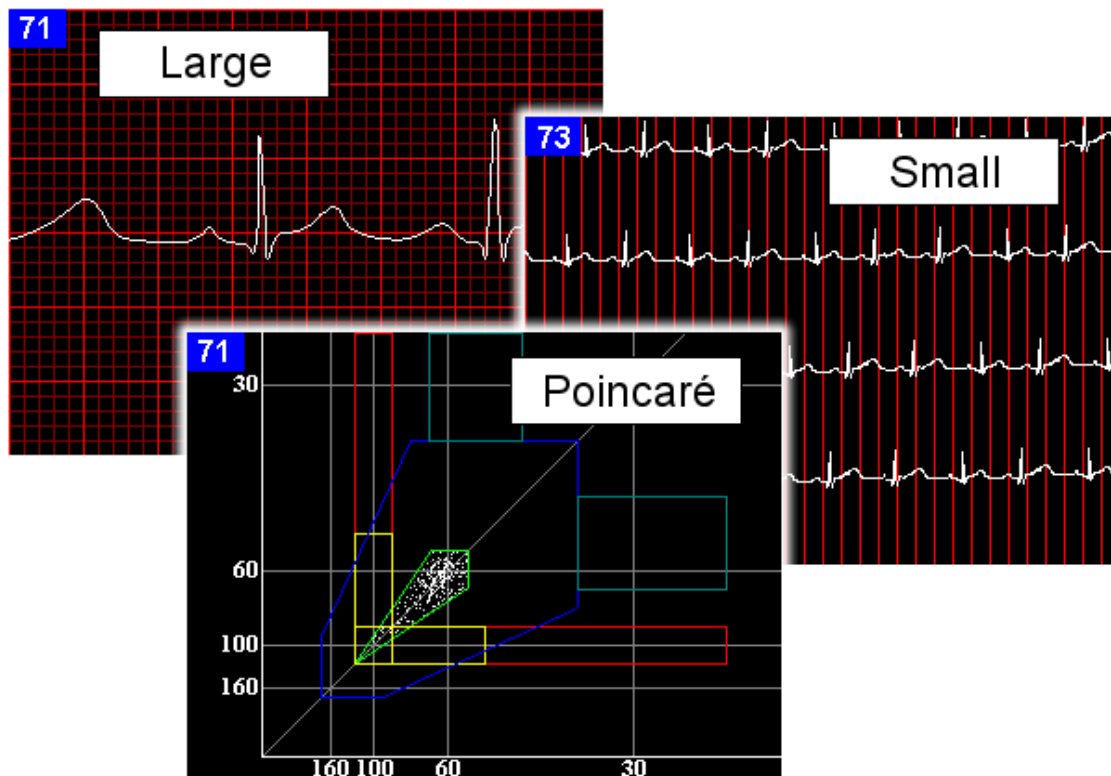
- Large display
- Small display
- Poincaré display

The large display allows you to examine one or two heart beats in detail.

The small display allows you to examine a couple of dozen heart beats to see the regularity of the rhythm.

The Poincaré display shows a history of the rhythm.

The following Steps briefly describe what you might look for in each display. Don't take my word for it. There are lots of pages on the web for teaching student doctors and nurses how to interpret ECGs. They're worth reading.



Step 7: Large Display

The large display mode allows you to examine one or two heart beats in detail. It shows a red background grid to make it look more like a ECG standard chart.

An ECG chart has large and small red squares printed on. The standard scale is

- the x-axis is time with one large square being 200mS
- the y-axis is voltage with one large square being 0.5mV.

Of course, the doctor could have fiddled with the knobs of the ECG machine and changed the gain or the chart speed but pictures in textbooks will show the standard scale.

The display shows the beats per minute in the top left corner. The sketch attempts to recognise "beats" but will be confused by noise or poor electrode connections.

The sketch tries to keep a peak in the left third of the screen so it isn't jumping around too much and is easier to study. It can only do so if the heart beat is regular but a healthy heart doesn't beat regularly so that feature doesn't work too well.

You should be able to see the classic ECG trace you get in a textbook.

The first bump is P wave. That's the atrial depolarisation - the muscles of the atria starting to contract. There's a pause - the "PR segment" - while blood flows into the ventricles.

The big spike is the QRS complex. That's the ventricular depolarisation - the muscles of the ventricles starting to contract. There's another pause while blood flows out of the ventricles.

The next bump is the T wave. That's ventricular repolarisation as the muscles of the ventricles relax. There's another pause and it all happens again.

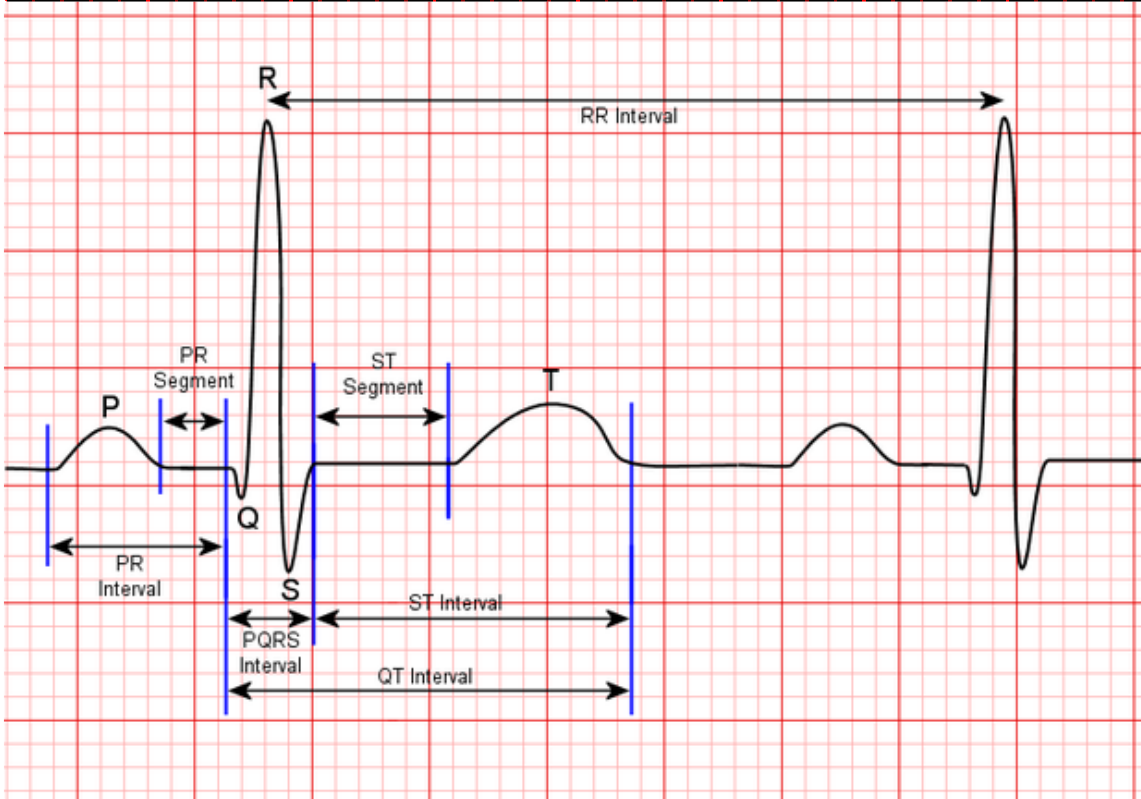
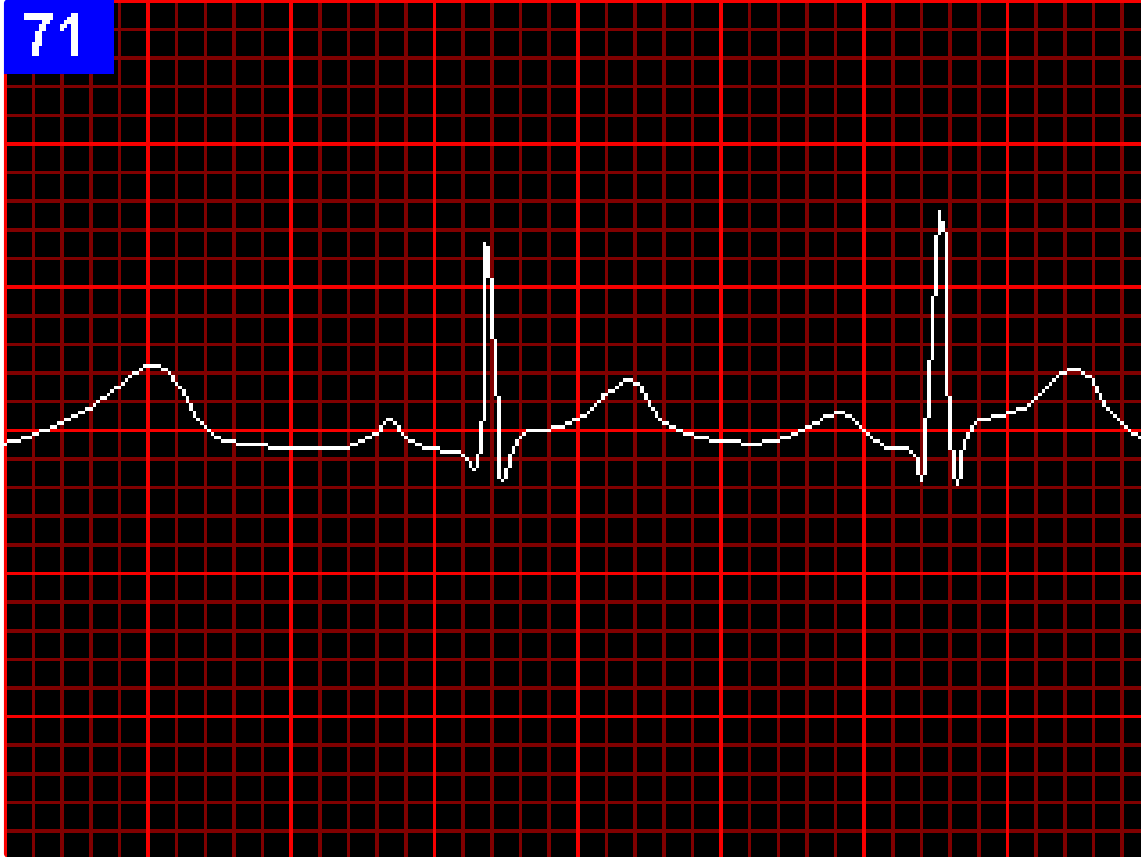
There ought to be a bump for atrial repolarisation but it's hidden under the QRS complex. There can be a U depolarisation wave but it's often too small to be seen.

It takes years to train a cardiologist but you should be able to see broadly what's going on. [Here](#), [here](#) and [here](#) are some web pages. Or search for "[ECG interpretation](#)".

The P wave can tell you about what's going on in the atria. For instance the shape of the P wave can tell you whether the left and right atria are acting together. If the P waves and the QRS waves are not synchronised then the two oscillators have become disconnected.

The QRS and ST segments show what's going on in the ventricles. An elevated ST segment can indicate a myocardial

infarction ("heart attack"). A depressed ST segment can indicate a myocardial ischaemia ("angina"). Both are due to parts of the ventricle muscles not functioning properly. Ischaemia is when a region of the muscles is struggling due to lack of oxygen; infarction is when a region is really in a bad way and has given up. But don't panic if you seem to have an elevated ST - it can also be due to the electrodes not being positioned properly.



Step 8: Small Display

The small display mode allows you to see the overall heart rate and look for irregularities. A "normal" heart rate depends on age

- 3-5 years: 80-120 bpm
- 6-10 years: 70-110 bpm
- 11-14 years: 60-105 bpm
- 15+ years: 60-100 bpm

The average woman's heart rate is 4bpm higher than a man's. Both male and female average heart rate drops by 4bpm as you get old.

"Bradycardia" is a heart rate below the normal range: less than 60bpm in adults. "Tachycardia" is a heart rate above the normal range: more than 100bpm in adults.

Clearly, the heart rate averaged over the last minute will depend on how hard you are exercising or how excited you are. The time between heart beats (the R-R interval) can also vary from beat to beat. One beat might be, for instance, 10% shorter than the next. That variability from beat to beat is called [HRV \(heart rate variability\)](#).

Higher HRV is considered to be a "good thing". Supposedly, the more variable the heart beat is, the better the heart is able to respond to different requirements. HRV decreases as heart rate increases (as you'll see in the next display). HRV also decreases with age, with infarction, with diabetes and all sorts of other conditions

"[Ectopic](#)" beats are an interval that is too short or too long. A major cause is something wrong with the conduction system. [Premature atrial](#), junctional and [ventricular](#) contractions can occur. I cannot find a definition of when a short or long interval is classed as ectopic rather than due to normal HRV but automatic systems often decide that any interval more that 20%-30% different from the preceding one are "ectopic". Ectopic beats sometimes have a different shape from normal ones.

Ectopic beats could be indications of something wrong. They could be due to damage from a heart attack, cardiomyopathy, valve malfunction, etc. Or they could be harmless perhaps caused by alcohol, caffeine, prescription drugs, stress, etc.



Step 9: Poincaré Display

The [Poincaré display](#) mode plots the length of one interval (from R to R) on the x-axis against the length of the next interval on the y-axis.

500 points are stored in a circular buffer. When the buffer is full, old points are overdrawn in black.

The axes are labelled in BPM.

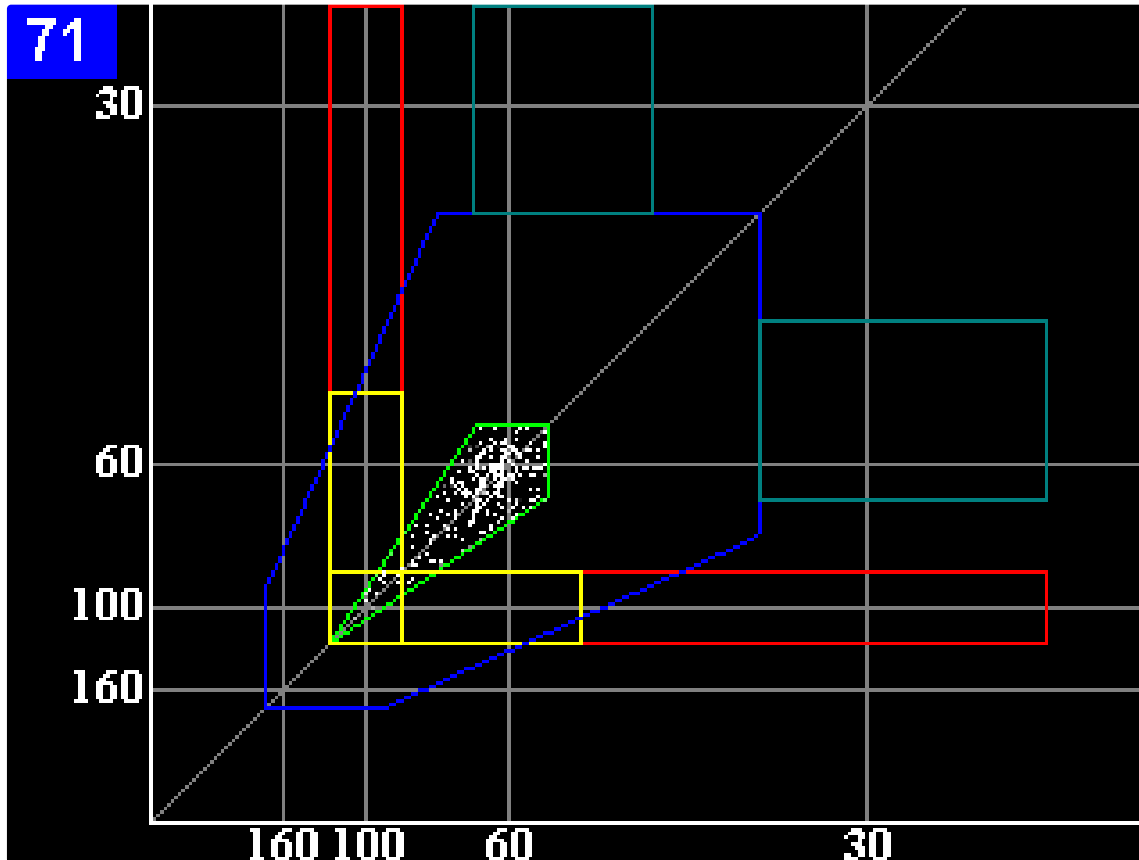
Some researchers think that HRV, ectopic beats and other irregularities in heart rate are best detected by a Poincaré display. Although ECG Poincaré plots been around since 1990, they are not "mainstream" and many workers prefer a simple chart.

If you google for "ecg poincare" you will find a lot of research papers.

[Here](#) is a good discussion of what can be seen in a Poincaré plot.

A normal Poincaré plot shows a diagonal band of points tapering to the bottom left. This demonstrates that HRV decreases as heart rate increases. An overall fatter band indicates greater HRV.

Tachycardia shows as a small group of dots near the lower left. Bradycardia shows as a scattered group towards the top right. Arrhythmia shows up as groups of dots away from the main diagonal. Fibrillation shows as a large cloud at the bottom left. A column of dots at the top left may be premature ventricular contractions. Pauses in beats show as clouds in the top-middle and right middle.



Step 10: Filtering

I found that the circuit had little noise so long as it was running on its own battery and the electrodes were making good contact. But there is some noise and I thought I'd try removing it with a digital filter.

You can read about digital filtering on an Arduino in my [Speech Recognition project](#). I won't repeat it all here. As the sample rate is very low - every 5mS - the Arduino has plenty of time to do the maths.

There is a good discussion of digital filtering [here](#) and an online calculator for the coefficients [here](#). The calculator gives the coefficients as real numbers; multiply them by 65536 before plugging them into my code.

Of course, you don't want the filter to mess up the shape of the "signal" - just to remove the "noise". It's important to look at the display with and without filtering and convince yourself you like the results. In particular, the Q and S parts of the wave have frequency components close to the "noise" frequencies you are trying to remove. You don't want the filter to alter them.

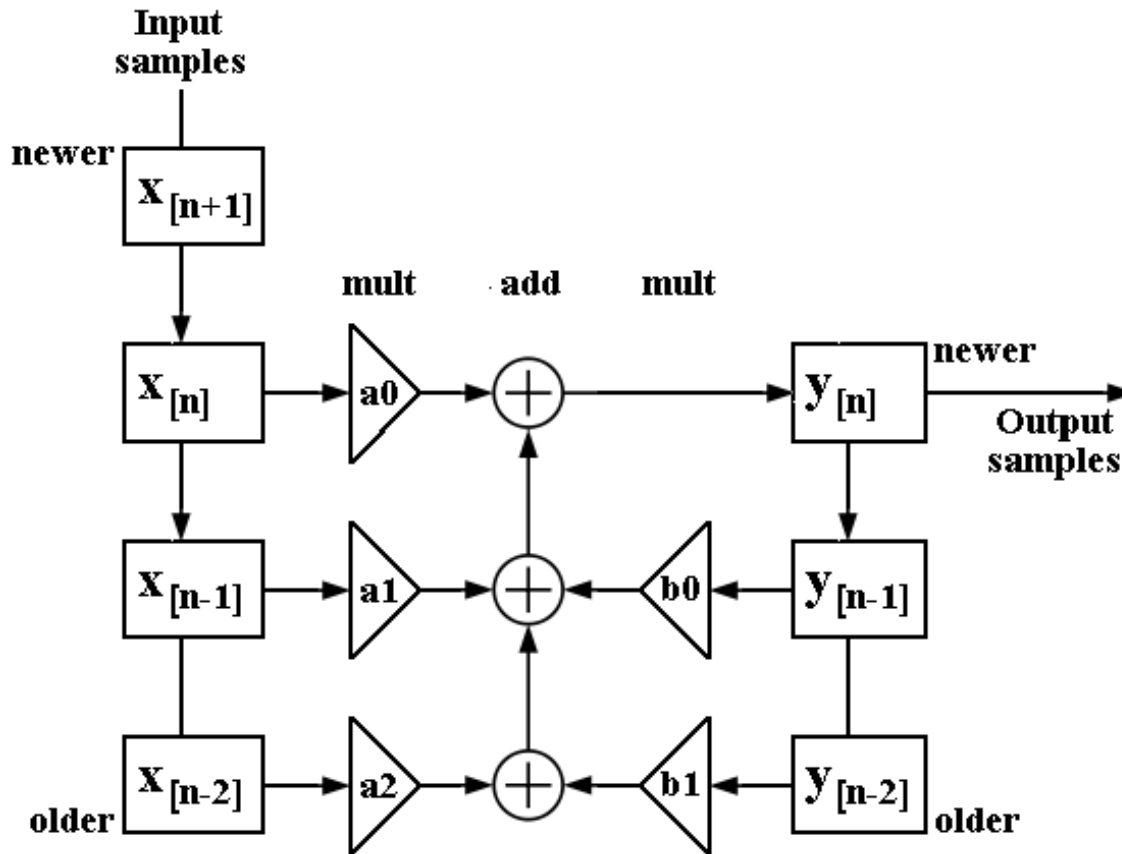
I tried a low-pass filter (40Hz Q=1) and it made some difference. The noise gets worse if you hold your hand near a mains device like a desk lamp. So I tried a 50Hz notch filter (Q=1 OR Q=2) - that removed a lot of noise. I liked the result and stuck with that. It seems that most of the noise is mains hum.

(In your country, mains hum might be at 60Hz. I've included a 60Hz notch filter as well.)

I have a philosophical distrust of filters. I think words like "signal" and "noise" can be a delusion. The "truth" is what you get from the ADC. Anything you do to the data after that is just you making the data prettier so it pleases you more.

I also wondered if a high-pass filter could remove the big fluctuations due to muscle movement. I added a high-pass filter with a cut-off frequency of 2Hz ($Q=1$). It may have fixed the big fluctuations but it also really messed up the P and T waves. I took it out.

You'll see calls to the different filter functions in the main loop() function. Comment them out (or in) as you see fit.

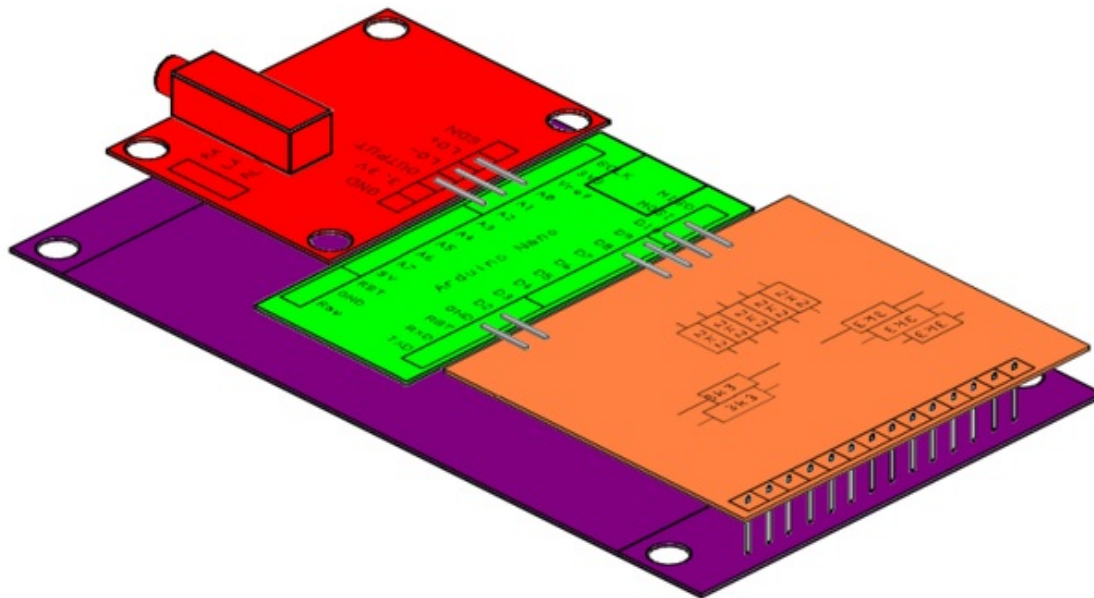
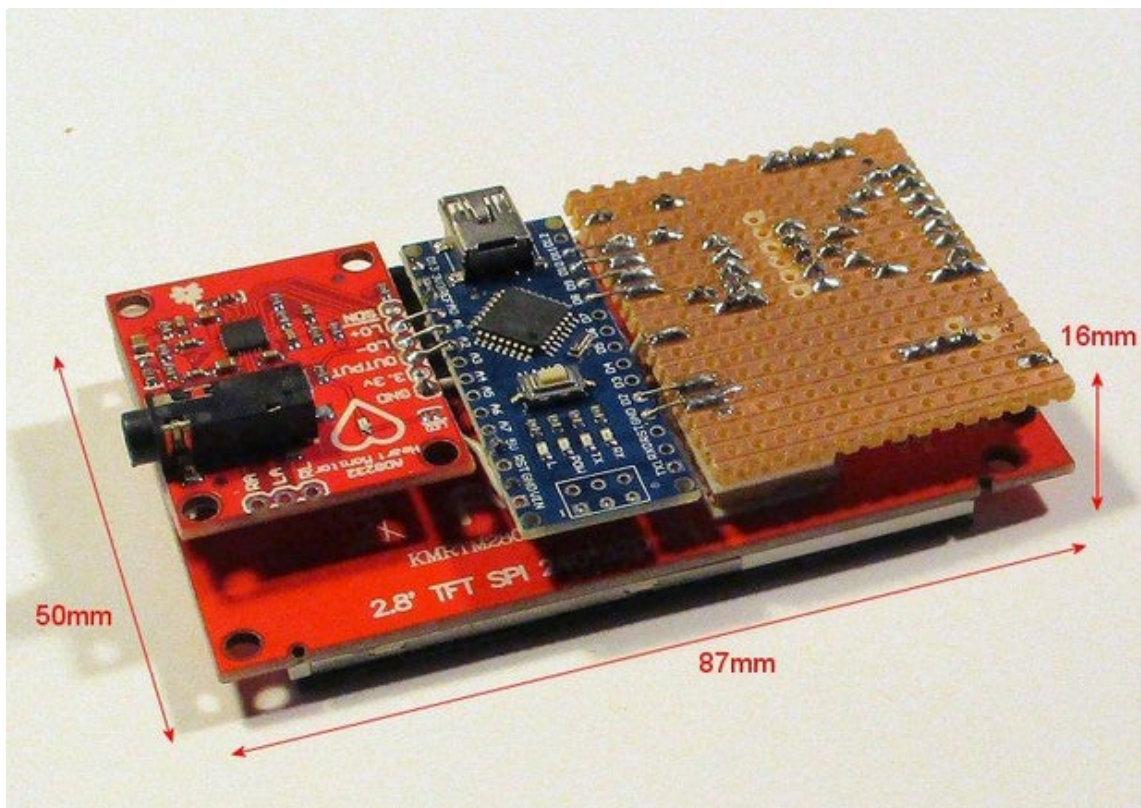


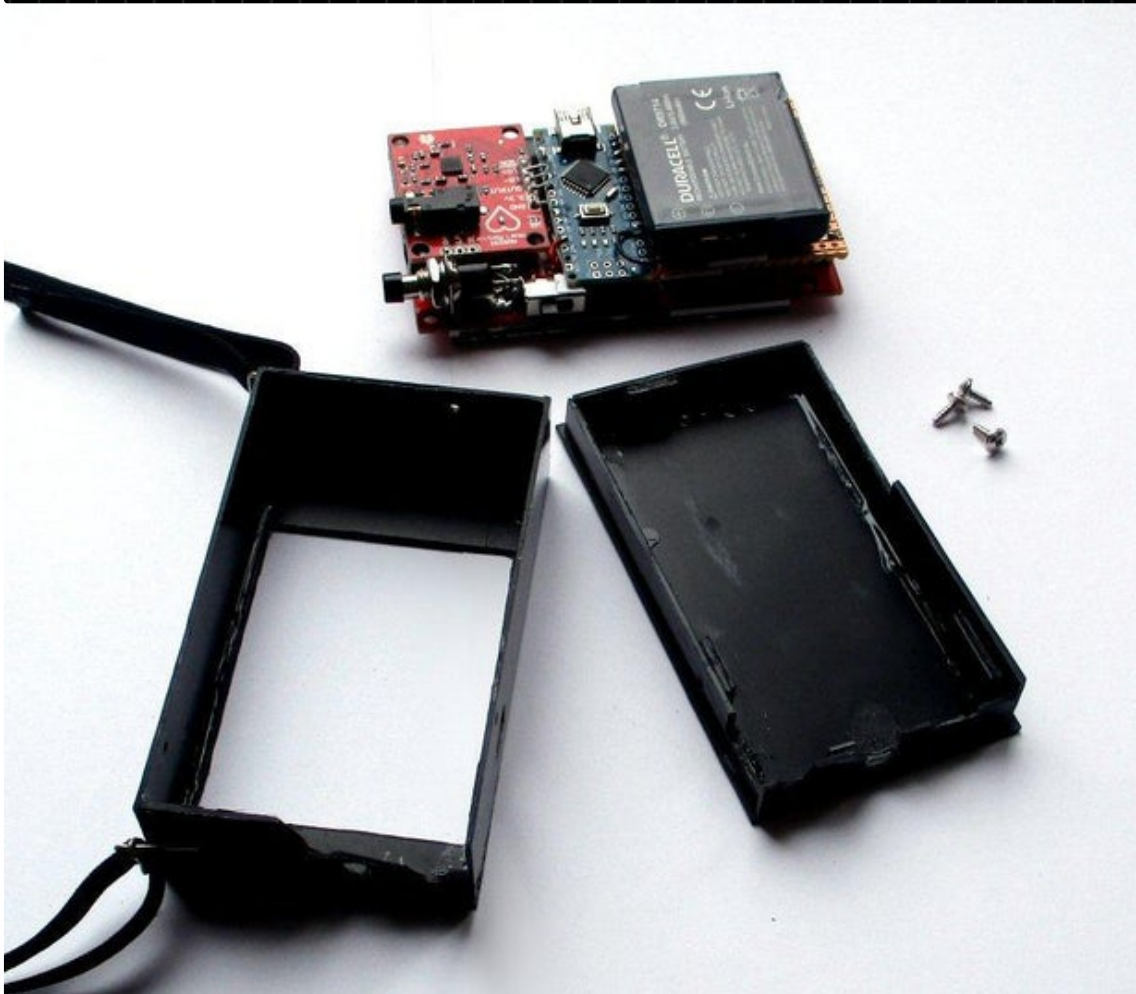
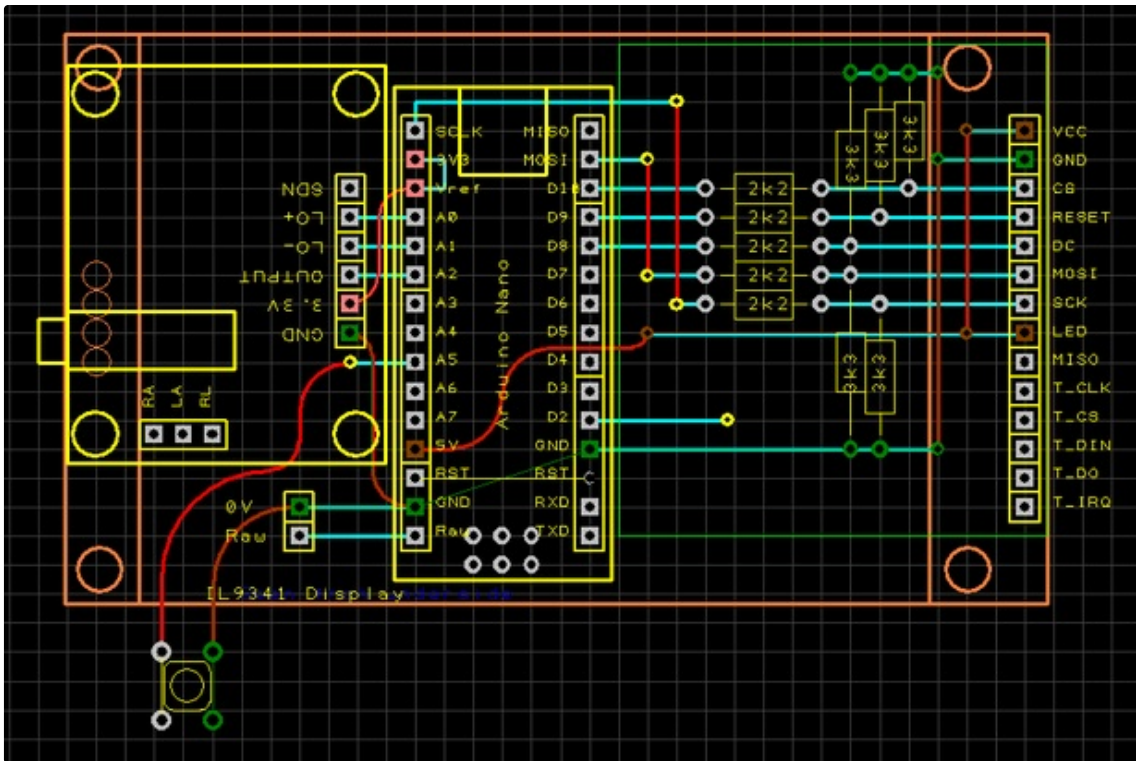
Step 11: Building It

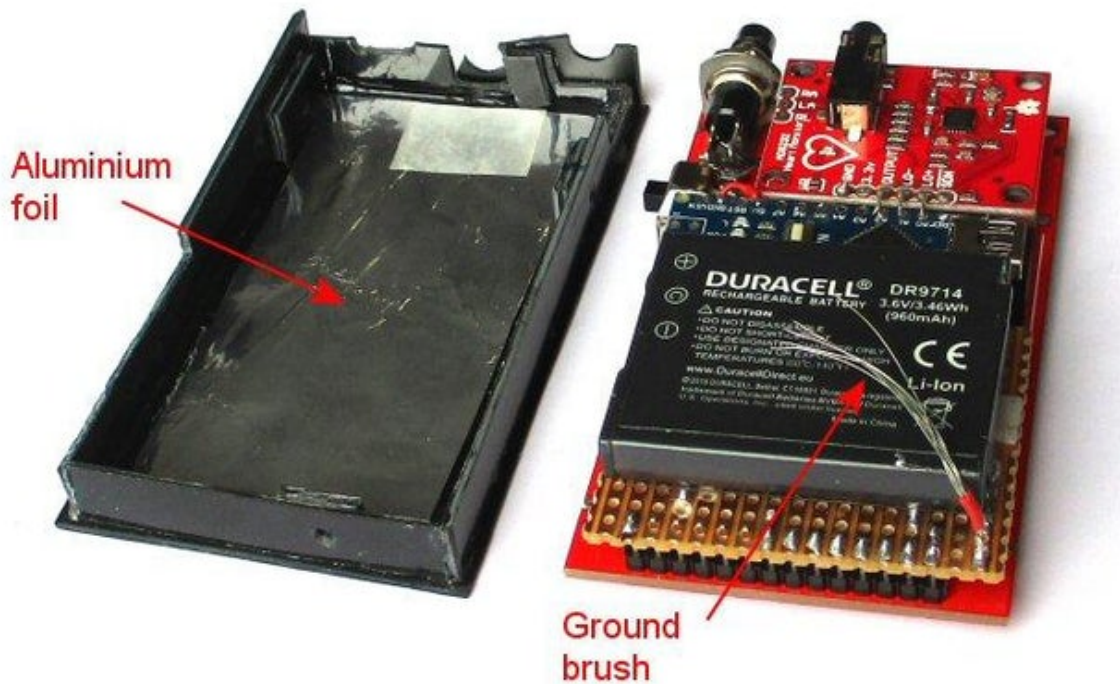
I put the voltage convertor resistors on a piece of stripboard and soldered it to the pins of the display. The AD8232 module is stuck onto the back of the display with sticky pads and a 2.5mm spacer. The Nano is suspended in between the two boards by soldered wire links. The whole assembly is quite compact and solid.

I thought about 3D printing a case but went for old-school methods. I glued 0.7mm polystyrene sheet together to make a box. The strap is old elastic that had lost its elasticity.

Shielding makes some difference to the noise level so I lined the back of the box with kitchen foil. A "brush" made from stripped cable connects it to ground







Step 12: Power Supply

Now it needs a battery. The total current consumption is around 127mA and I'd like it to run for 8 hours so that's 1000mAh.

Surprisingly, the Arduino, the AD8232 module and the display all run fine with 3.7V on the "5V" pin. By the time it gets down to 3V, the display is getting dim but everything still works.

My first thought was to run it 4 AA cells. An AA cell has a capacity of over 1000mAh. The initial voltage of 4 typical AA cells is more than 6V but it drops to 4.8V as it discharges. The Arduino cannot withstand 6V on its "5V" pin so we must connect the 4 AA battery pack to the Arduino's Vin pin. The Arduino has a regulator with a dropout of around 0.7V so the voltage of the Arduino's "5V" pin will be above 4V for most of the battery life.

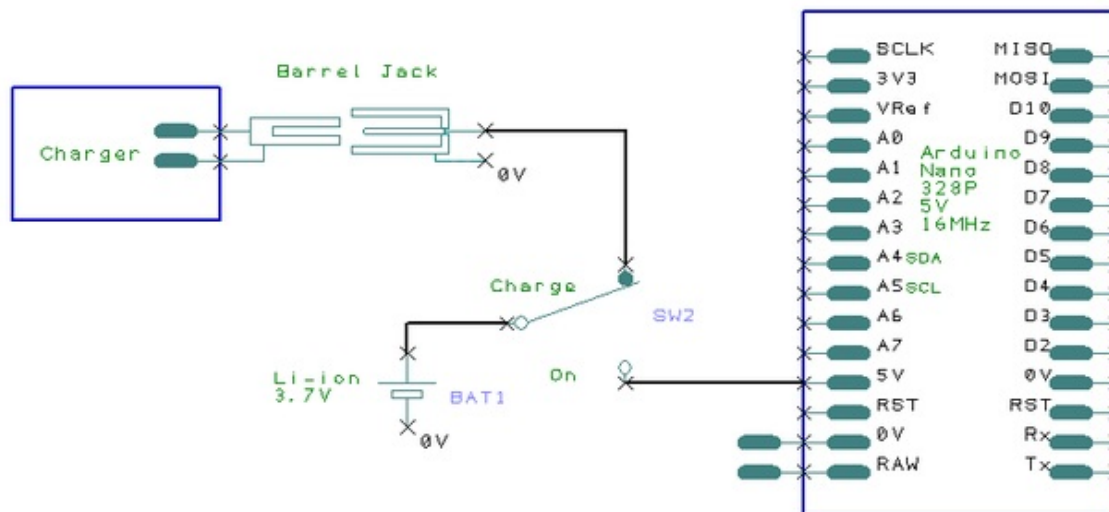
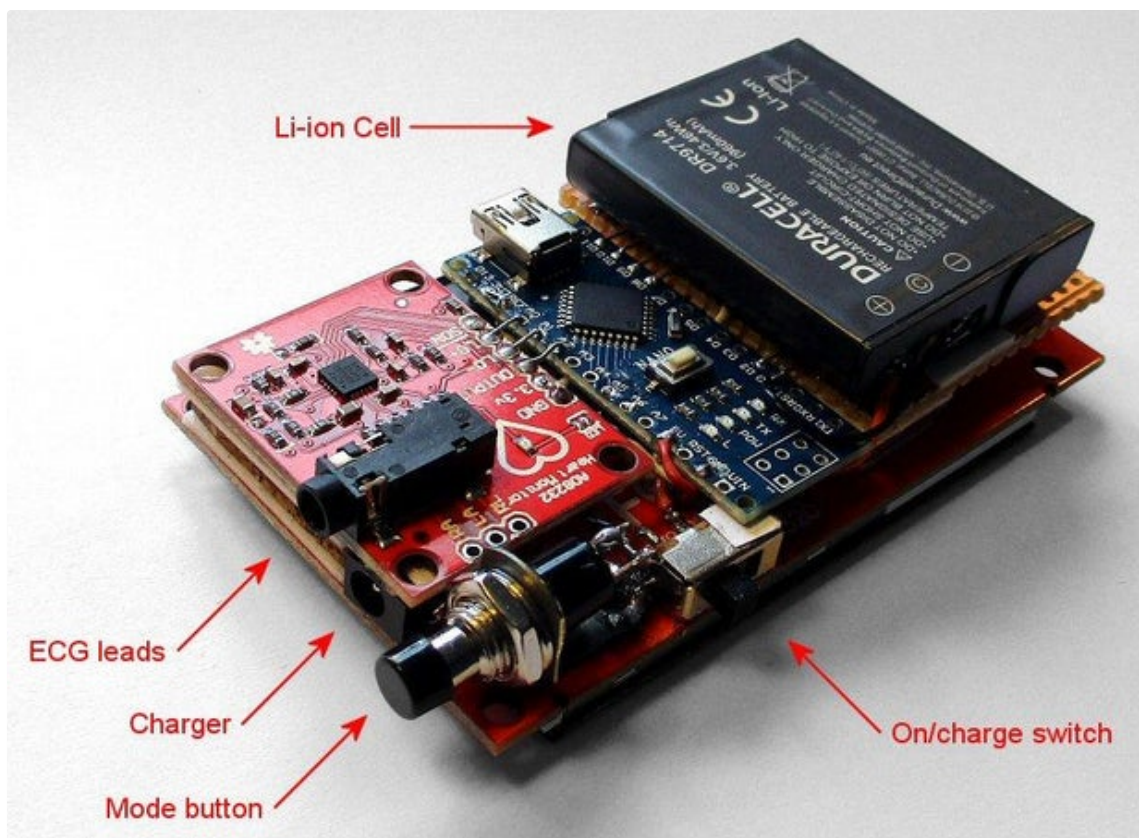
AA cells are a good choice. They have the advantage that "recharging" takes a minute. The only problem is their huge size. A nice slim unit would have a big fat battery pack on the back.

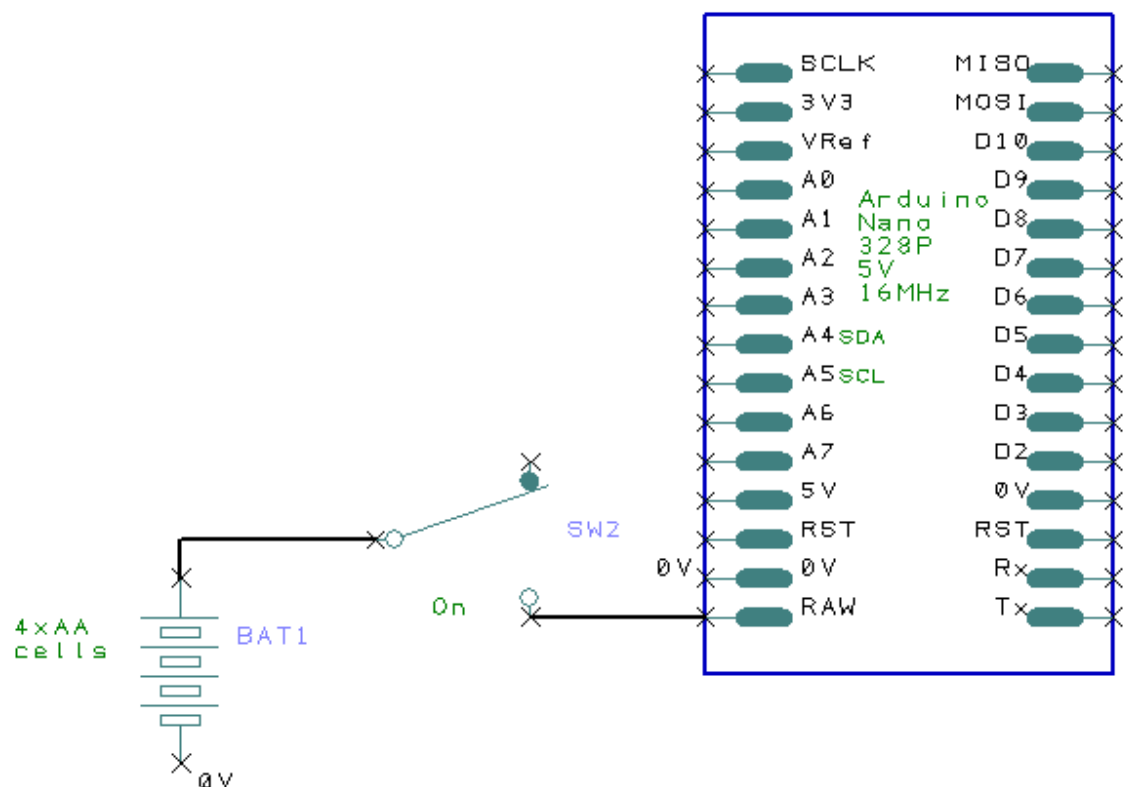
When a local electronics shop closed I bought a big bag of Li-ion battery packs at a good price. It included several NP-BG1 packs (for Sony Cyber-shot). An NP-BG1 contains a 1000mAh Li-ion cell (capacity depends on manufacturer).

What if we connect a 3.7V Li-ion cell to the "5V" pin of the Arduino? It all works fine. That simplifies the power supply. So I added a Li-ion cell and a 3.4mm barrel jack to charge it.

(The NP-BG1 contains a thermistor to check for overheating so a 3-pin connector would have been better. But the thermistor is the wrong value for the charger I'm using and I did without.)

The On/Off switch connects the cell to either the Arduino or the charger so it's impossible to charge the unit when it's in use.





Step 13: Future

How can this project be expanded?

It would be nice to save a whole day's worth of data. With a sample period of 5mS, that's 17Mbytes of data. At first sight, an SD card would be ideal but the write time of an SD card is very variable. Sometimes it takes a couple of mS but sometimes it erases a large number of blocks of flash memory and takes couple of hundred mS. An SD card can cope with the average data rate so I'd probably have a second Arduino that buffered the data stream from the ECG Arduino and wrote it occasionally in large blocks - I wouldn't trust multi-tasking on a single Arduino but it might work. Serial EEPROMs are available that can write fast enough but only hold a few Mbits. A large flash memory chip may be the way to go - you may be able to erase it all (a slow operation) at the start of the day then write individual bytes in a few mS. But an SD card has the advantage that you can take it out and plug it into a PC.

There are a ridiculous number of [ECG file formats](#). The [HL7 aECG](#) standard might be the one to go for.

If you stored a whole day's data then the Poincaré plot becomes really useful. Click on a point and that beat is displayed so you can quickly find weird beats.

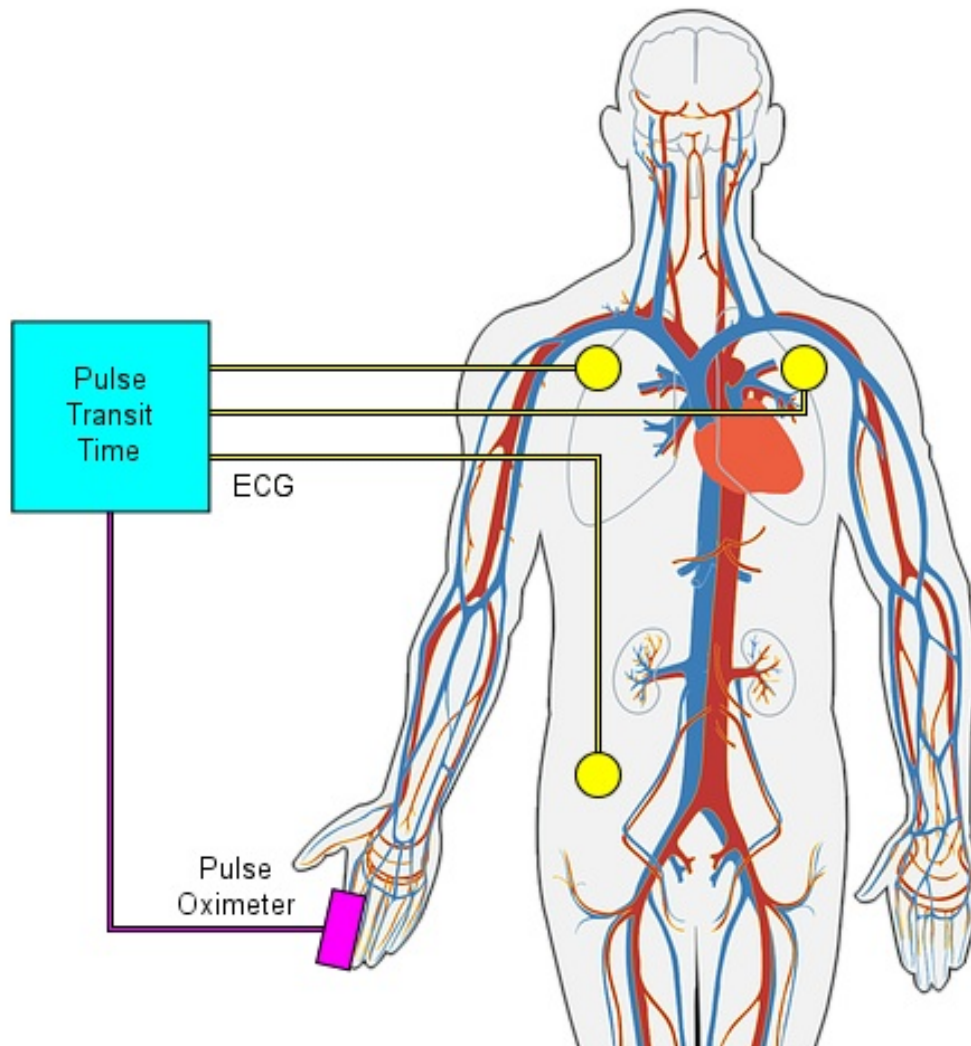
[Pulse arrival time](#) (PAT) or pulse transit time (PTT) is the time interval between a R wave in the ECG and the measurement of the pulse on (e.g.) your finger with a pulseoximeter. Pulseoximeters are very easy to build. It seems that [pulse arrival time is related to blood pressure](#). It needs calibrating for each individual but then it's a simple non-invasive way of [measuring blood pressure](#) over a whole day. It can also be used to estimate arterial stiffness. There is a [linear relationship](#) between pulse transit velocity and systolic blood pressure.

Can the ECG act as an [electro-oculograph](#)? I taped the ECG electrodes just to the outside of my left and right eye sockets. It worked. I could easily measure a 0.7mV "square wave" as I moved my eyes to the left or right. The problem is that the ECG module I'm using has built-in filters. The high-pass filter removes signals below about 5Hz so it a slow square wave looks like [this](#).

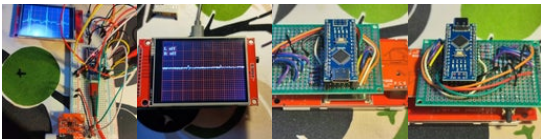
Maybe using the AD8232 chip without the rest of the module would mean a different filter could be used but the AD8232 has lots of other built-in signal processing. I think it would be worth trying if you're interested in electrooculography.

Taping the electrodes above and below the eye didn't work at all well. I could hardly see an effect. But a blink was easily visible.

I also tried to get electromyograms using the ECG machine. It didn't work. I think myograms should be regarded as fast AC signals. Electromyogram circuits generally include a rectifier and smoother so the overall AC "energy" of the signal is measured. The filtering in the AD8232 module removes all those high frequencies and myograms don't show up at all.



What a great project and very nice support from the author Peter - thanks again! First build on Breadboard, then on a PCB, very compact design, I mounted the Arduino and TFT on headers so I can use them on later projects easy and change if needed. Still very compact and very suitable for a small housing I might do later. My TFT was very sensitive to the power input. As Peter mentioned there are different versions, I connected VCC to the 3.3V pin and the LED to 5V, works perfect.



Thanks.

I used to use headers so I could reuse components in later projects then I realised that Arduinos and displays and whatnot are so cheap I should just treat them as consumables. A new project means a brand new Arduino and display. It feels immoral somehow - it's a computer and computers are valuable, right? The first computer I programmed was about as powerful as an Arduino; it filled a room and weighed 10 tons.

Hi,
I've copied the 3 files in a new folder on my desktop; Openned ArdECG1 in Arduino 1.8.13 IDE; Ran compil and got many errors like these ones:

```
*****
C:\Users\bill\ Desktop\ArdECG1\ArdECG1.ino: In function 'void DrawGridPoincare()':
ArdECG1:160:40: error: invalid conversion from 'const byte* {aka const unsigned char*}' to 'word
{aka unsigned int}' [-fpermissive]
DrawInt(f[i], MediumFont, TFT_WHITE);
```

^

In file included from C:\Users\billa\Desktop\ArdECG1\ArdECG1.ino:8:0:
sketch\SimpleILI9341.h:54:6: note: initializing argument 2 of 'void DrawInt(int, word, uint16_t)'
void DrawInt(int i, word Font, uint16_t color);

^~~~~~

ArdECG1:162:40: error: invalid conversion from 'const byte* {aka const unsigned char*}' to 'word
{aka unsigned int}' [-fpermissive]
DrawInt(ff[i], MediumFont, TFT_WHITE);

^

In file included from C:\Users\billa\Desktop\ArdECG1\ArdECG1.ino:8:0:
sketch\SimpleILI9341.h:54:6: note: initializing argument 2 of 'void DrawInt(int, word, uint16_t)'
void DrawInt(int i, word Font, uint16_t color);

.....

PS: I use MiniCore boards library (compil OKay with AVR boards libray)
One idea ?



Are these errors all to do with using fonts?

The error "invalid conversion from 'const byte*' to 'word' in 'DrawInt(ff[i], MediumFont, TFT_WHITE);'" might be to do with the 'MediumFont' parameter.

I've implemented Fonts in a weird way that could be unique to Arduinos.

So as to save RAM space, constant arrays (such as fonts) can be held in ROM. They are declared as e.g.

```
const byte MediumFont[] PROGMEM = {  
  15, // ymax  
  4, // descender  
  ...
```

note the PROGMEM construct. You can google for how PROGMEM is implemented and what it means.

You have to use special code "pgm_read_byte_near" to access data stored in a PROGMEM array. For instance...

```
void DrawInt(int i, word Font, uint16_t color)
```

calls

```
void DrawChar(uint8_t c, word Font, uint16_t color)
```

which uses the construction (e.g.)

```
b = pgm_read_byte_near(Font);
```

Could it be that the MiniCore boards library has some other way of storing constant arrays in ROM?

Peter



I was on the way but did not went up to the ROM and PROGMEM ;
The fact is, depending on how one burn the minicore bootloader, EEPROM can be retained or not; Is the problem located in that option ?
Anyway, I re-burned the pro-mini I used with the standard Bootloader to make your prgm running... and it is running properly.

I did not succeed in viewing my ECG, probably because I use a 128 x 128 display ST7735 based on; but I will go on tomorrow with the fake pulse for testing and will keep you advised..(the 3 grids are OKay, the L-OFF and R off are mirrored if electrodes are disconnected, hart beat led blinking and no trace...may be this is due to ST7735 in my LCD instead of a ILI9341)



Hello, I have exactly the same issue - many errors when compiling,
"warning: invalid conversion from 'const byte* {aka const unsigned char*}'" and others. I have the AVR Arduino Boards 1.8.5 Library installed - before 1.6 also not working.
I could upload it to the arduino but no trace on the screen - L-R detection is working fine also

shown on the TFT screen and the led on the control board is blinking acc. to the fake ECG. I also try the ArdECG0.ino and check the serial output - always shows 1023 - I used some other ECG test program that I found and could easily get a ECG output on the serial monitor and plotter. Is there some issue addressing the input? I am using Arduino 1.8.19, and also tried to downgrade to the .9 version but same errors and results.



Try turning off Warnings. In the File|Preferences dialog set Compiler Warnings to None. I reckon that the warnings that GCC gives are fairly useless - they're telling me things that really, really don't matter.

I don't see any errors in the screenshot.

I'm using version 1.8.9 with Compiler Warnings to set None.

If you're not getting any errors then it should run OK.

If ArdECG0.ino is always returning 1023 then there's probably a problem in the hardware so that the input into to pin A0 is always high. Can you look at it with an oscilloscope or meter?



Hello Peter, I just build up all again to avoid any mistakes. - All is working fine now!

I guess I made some wiring mistake. The compile "errors" or better warning still exists but no problems at all.

I found my ECG Module has some issues - seems to have some cold soldering somewhere.

One other point that maybe helps others: I have exactly the same TFT module with touch function V1.2 but mine seems very sensitive, if I connect it to the 5V it sometimes just shuts itself off, but it's working perfect on the 3.3V output.



How did you find the different leads with only 3 electrodes? Can you please explain...



As I say in Step 4 "what an ECG displays is the difference between two electrodes - it's a differential amplifier. The LL or RL electrode acts as a ground to help with common-mode rejection". What a doctor calls a "Lead" is not an electrode.

In Step 4 I explain how different pairs of electrodes give different "Leads".

As the ECG in this instructable only contains a single differential amplifier, it can only display one "Lead" at a time.

By moving the electrodes around your can see different "Leads".

"Lead aVR" is RA - average of (LA,LL). So you could produce the average of (LA,LL) by using a couple of resistors.

Figure 21 of this webpage shows what different "Leads" should look like.

<https://ecgwaves.com/topic/ekg-ecg-leads-electrodes-systems-limb-chest-precordial/>

Peter



Thank you for ur reply. I made an ECG device using AD8232 and Arduino. I also used multiplexers for switching between different electrodes for different lead combinations (eg : for 1st 5 seconds it measure LA-RA ie:- Lead I, for next 5 seconds it will measure LL-RA:- Lead II. Like that we can get the 3 lead combinations. From this, can I calculate aVR , aVL, and aVF?

ps: if all the leads are measuring at a time, it is easy to calculate aVR , aVL, and aVF



In the reply to mwhittle below, I discuss how to use resistors to create the different "average" values.



the connections as described, are not correct:

LA Left Arm: Green ->>> This should be Yellow

RA Right Arm: Red ->>> Correct

RL Right Leg: Yellow ->>> This should be Green

If connected the right way, it works perfect!

(thanks to my wife, who is cardio-nurse)



It seems that there is no agreed standard.

As I said in Step 2 "With the module I bought, ... Yours may be coloured differently". And I explained how to tell which is which.

The AHA (American Heart Association) says:

Right Arm White
Left Arm Black
Left Leg Red

But the IEC (International Electrotechnical Commission) agrees with your wife:

Right Arm Red
Left Arm Yellow
Left Leg Green

Neither agrees with what the Chinese company sent me!

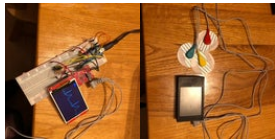
I'm glad you got yours working. It looks good in the photo.



Ok, multiple standards, "how nice". We did some measurements, with Yellow on the Leg gives a wrong ECG with a strange high P-top, with green on the leg, the ECG was right, so it is also possible that there are different cables on the market, that means that for other people Yellow on the leg is right



Works Perfect, nice job! Also found a nice small case for it :
<https://www.thingiverse.com/thing:3026549/files>



Hi All This is a nice project!!!
I get a lot of compilation errors, running OSX 12.1 (M1) and Arduino 1.8.16 and a Arduino-Nano 328
any Idea what this can be?

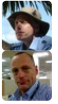
```
Arduino: 1.8.16 (Mac OS X), Board: "Arduino Nano, ATmega328P (Old Bootloader)"
/var/folders/x6/w4vl81fd7qg547mqgf792mq80000gp/T/ccKeEyxS.ltrans0.ltrans.o: In function
`calcBPM(unsigned char, unsigned int)':
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:302: undefined reference to
`DrawBox(unsigned int, unsigned int, unsigned int, unsigned int, unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:303: undefined reference to
`ILI9341SetCursor(unsigned int, unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:304: undefined reference to `DrawInt(int,
unsigned int, unsigned int)'
/var/folders/x6/w4vl81fd7qg547mqgf792mq80000gp/T/ccKeEyxS.ltrans0.ltrans.o: In function
`DrawPoincare':
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:265: undefined reference to
`DrawPixel(unsigned int, unsigned int, unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:268: undefined reference to
`DrawPixel(unsigned int, unsigned int, unsigned int)'
/var/folders/x6/w4vl81fd7qg547mqgf792mq80000gp/T/ccKeEyxS.ltrans0.ltrans.o: In function
`DrawPoincareLine(int, int, int, int, unsigned int)':
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:125: undefined reference to
`DrawLine(unsigned int, unsigned int, unsigned int, unsigned int, unsigned int)'
/var/folders/x6/w4vl81fd7qg547mqgf792mq80000gp/T/ccKeEyxS.ltrans0.ltrans.o: In function
`DrawGridLarge':
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:92: undefined reference to
`ClearDisplay(unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:95: undefined reference to
`DrawHLine(unsigned int, unsigned int, unsigned int, unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:98: undefined reference to
`DrawVLine(unsigned int, unsigned int, unsigned int, unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:102: undefined reference to
`DrawHLine(unsigned int, unsigned int, unsigned int, unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:105: undefined reference to
`DrawVLine(unsigned int, unsigned int, unsigned int, unsigned int)'
/var/folders/x6/w4vl81fd7qg547mqgf792mq80000gp/T/ccKeEyxS.ltrans0.ltrans.o: In function
`DrawGridSmall':
```

```

/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:114: undefined reference to
`ClearDisplay(unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:117: undefined reference to
`DrawVLine(unsigned int, unsigned int, unsigned int, unsigned int)'
/var/folders/x6/w4vl81fd7qg547mqgf792mq80000gp/T/ccKeEyxS.ltrans0.ltrans.o: In function
`DrawGridPoincare':
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:153: undefined reference to
`ClearDisplay(unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:157: undefined reference to
`DrawVLine(unsigned int, unsigned int, unsigned int, unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:158: undefined reference to
`DrawHLine(unsigned int, unsigned int, unsigned int, unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:159: undefined reference to
`ILI9341SetCursor(unsigned int, unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:160: undefined reference to `DrawInt(int,
unsigned int, unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:161: undefined reference to
`ILI9341SetCursor(unsigned int, unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:162: undefined reference to `DrawInt(int,
unsigned int, unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:164: undefined reference to
`DrawLine(unsigned int, unsigned int, unsigned int, unsigned int, unsigned int)'
/var/folders/x6/w4vl81fd7qg547mqgf792mq80000gp/T/ccKeEyxS.ltrans0.ltrans.o: In function
`DrawPoincareLine':
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:125: undefined reference to
`DrawLine(unsigned int, unsigned int, unsigned int, unsigned int, unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:125: undefined reference to
`DrawLine(unsigned int, unsigned int, unsigned int, unsigned int, unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:125: undefined reference to
`DrawLine(unsigned int, unsigned int, unsigned int, unsigned int, unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:125: undefined reference to
`DrawLine(unsigned int, unsigned int, unsigned int, unsigned int, unsigned int)'
/var/folders/x6/w4vl81fd7qg547mqgf792mq80000gp/T/ccKeEyxS.ltrans0.ltrans.o: Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:125:
more undefined references to `DrawLine(unsigned int, unsigned int, unsigned int, unsigned int,
unsigned int)' follow
/var/folders/x6/w4vl81fd7qg547mqgf792mq80000gp/T/ccKeEyxS.ltrans0.ltrans.o: In function
`DrawGridPoincare':
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:183: undefined reference to
`DrawVLine(unsigned int, unsigned int, unsigned int, unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:184: undefined reference to
`DrawHLine(unsigned int, unsigned int, unsigned int, unsigned int)'
/var/folders/x6/w4vl81fd7qg547mqgf792mq80000gp/T/ccKeEyxS.ltrans0.ltrans.o: In function
`CheckLead(unsigned char, int, char*, bool*)':
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:432: undefined reference to
`DrawBox(unsigned int, unsigned int, unsigned int, unsigned int, unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:433: undefined reference to
`DrawStringAt(int, int, char*, unsigned int, unsigned int)'
/var/folders/x6/w4vl81fd7qg547mqgf792mq80000gp/T/ccKeEyxS.ltrans0.ltrans.o: In function
`setup':
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:629: undefined reference to
`ILI9341Begin(unsigned char, unsigned char, unsigned char, unsigned int, unsigned int, unsigned
char)'
/var/folders/x6/w4vl81fd7qg547mqgf792mq80000gp/T/ccKeEyxS.ltrans0.ltrans.o: In function
`DrawGridVLine':
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:200: undefined reference to
`DrawVLine(unsigned int, unsigned int, unsigned int, unsigned int)'
/var/folders/x6/w4vl81fd7qg547mqgf792mq80000gp/T/ccKeEyxS.ltrans0.ltrans.o: In function
`DrawTraceLarge':
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:342: undefined reference to
`DrawVLine(unsigned int, unsigned int, unsigned int, unsigned int)'
/var/folders/x6/w4vl81fd7qg547mqgf792mq80000gp/T/ccKeEyxS.ltrans0.ltrans.o: In function
`DrawGridVLine':
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:202: undefined reference to
`DrawVLine(unsigned int, unsigned int, unsigned int, unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:207: undefined reference to
`DrawPixel(unsigned int, unsigned int, unsigned int)'
/var/folders/x6/w4vl81fd7qg547mqgf792mq80000gp/T/ccKeEyxS.ltrans0.ltrans.o: In function

```

`DrawTraceSmall':
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:375: undefined reference to
`DrawVLine(unsigned int, unsigned int, unsigned int, unsigned int)'
/Users/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:381: undefined reference to
`DrawVLine(unsigned int, unsigned int, unsigned int, unsigned int)'
collect2: error: ld returned 1 exit status
exit status 1
Error compiling for board Arduino Nano.
This report would have more information with
"Show verbose output during compilation"
option enabled in File -> Preferences.



I looks like one of the libraries is missing.

Hi Peter

I don't get errors about missing lib's, tried a clean install on a different pc with Ubuntu and arduino 1.8.6

(for info ArdECG0 is running fine, problem is with ArdECG1)
getting the same errors, first I get a lot of warning,
no compiling errors, but when linking together I get :

Linking everything together...

```
/home/roland/.arduino15/packages/arduino/tools/avr-gcc/5.4.0-atmel3.6.1-arduino2/bin/avr-gcc -  
Wall -Wextra -Os -g -flto -fuse-linker-plugin -Wl,--gc-sections -mmcu=atmega328p -o  
/tmp/arduino_build_959534/ArdECG1.ino.elf  
/tmp/arduino_build_959534/sketch/ArdECG1.ino.cpp.o  
/tmp/arduino_build_959534/libraries/SPI/SPI.cpp.o /tmp/arduino_build_959534/core/core.a -  
L/tmp/arduino_build_959534 -lm
```

```
/tmp/cc3jVZjJ.ltrans0.ltrans.o: In function `calcBPM(unsigned char, unsigned int)':  
/home/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:304: undefined reference to  
`DrawBox(unsigned int, unsigned int, unsigned int, unsigned int, unsigned int)'  
/home/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:305: undefined reference to  
`ILI9341SetCursor(unsigned int, unsigned int)'  
/home/roland/Desktop/ECG/ArdECG1/ArdECG1.ino:306: undefined reference to `DrawInt(int,  
unsigned int, unsigned int)'  
**** and a lot more ****
```



DrawBox, ILI9341SetCursor and so on are in the SimpleILI9341 library which can be downloaded in Step 5.



Found the problem, quite stupid, if I download the SimpleILI9341.cpp file, for some strange reason my MacBook decide to save it as .cp in stead of .cpp
renaming the file to cpp and it works geat!



I like the design and am giving it some thought. I just finished my second episode of atrial fibrillation--the sections of my heart don't work in the right synchronization. It would be interesting to see when another episode is coming.

I believe your design is likely safe. I believe it enough I would not be reluctant to try it and will read through the design to see what components I don't already have.

However, your thought about SAFETY--"It's running off 4 AA cells or a single Lithium cell for goodness sake - how dangerous can it be?" Back before dirt was invented, I had my first electronics class. The professor pointed out that a circuit using only a few flashlight cells could be fatal with only connecting wires added. For clarification, but not instruction, it wouldn't be quick and would be awfully unpleasant, but dead is dead and in the right circumstances could be the result.



> atrial fibrillation ... It would be interesting to see when another episode is coming.

Does that mean 24-hour monitoring? Software to spot the symptoms? An interesting project.

> "It's running off 4 AA cells or a single Lithium cell for goodness sake - how dangerous can it be?"

Somewhere in the comments, someone talked about a person being injured by a multimeter. It's

hard to imagine. Apparently they poked the prongs through their skin. I guess skin resistance is the main barrier to hurting yourself with a small battery. How many garage mechanics touch 12V wires every day without noticing?

I have discussions with one of my clients who'll say things like "what if a child jumps across the desk and licks the microphone?".

I have an old ECG machine from the 1960s. There's no way that would pass muster today. Perhaps we were all tougher back then.

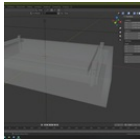
Peter



This is very cool well done. Will definitely be adding to my list of things to build :)



I made this and it works great. I also soldered to D2 and didn't know why the switch did nothing until I remembered someone posted a correction to A5. Once I moved that wire it all worked great. I also integrated a Lipo/Li-ion Micro USB charger instead of the barrel jack. I am now designing a 3D printer case with holes for the buttons and switches.



Hi Michael. Do you think you can share your 3D-printed design with a .stl file of it.
Best regards,
Kjell Nilsson



I haven't completed the design yet since I have been quite busy but as soon as I do I will send it up.



An integrated USB charger would be much better than what I did. I was in a hurry to finish the project.

Is there still a mistake in the instructable about the pushbutton? I thought I'd fixed that.



In step 11 photo 3 of the circuit diagram it's still going to pin D2.



OK. I've fixed it. Thanks.



Collected in a box.



That looks good. what battery did you use?

Rubber bulb electrodes eh? How well do they work? How long before they fall off.

I have an ancient ECG from the 1950s (from a car-boot sale) with bulb electrodes. I ought to try them.



I want to thank you again for the development and firmware. At my age, this device became necessary for me, because arrhythmia appeared. I use NCR18650B rechargeable battery, charger module and 5V converter on MC34063 to power the indicator. Rubber pears hold up perfectly. If you moisten the body with a saline solution, then they last indefinitely and give good contact. Of course, if the pears are of good quality.)

I have a question. Is it possible to supplement the button for recording a frame 'STOP-FRAME', so that it is convenient to read the amplitude and intervals between PQRSTs? In dynamic mode, this is not very convenient.

Sorry for the bad english.



That's a good idea - a second button for "freeze".

Are you able to write Arduino C programs?

I've made some changes to ArdECG1.ino to give ArdECG2.ino. I don't have time to test it now as I'm in the middle of another project but maybe you could test it.

In ArdECG2.ino there's a second button which connects A4 to ground. When the button is pressed, Stopping is set to true. When the display reaches the end of the display line, Stopped is

set true. Search the sketch for "BUTTON_STOP", "Stopping" and "Stopped" to see how they're used.

I've attached a file to this Reply but I don't know whether Instructables allows attachments here.



Hi Peter!

I tested ArdECG2.ino but the program doesn't work yet. Pressing the STOP button does not react in any way to stopping the cycle and recording. If you have any thoughts and other solutions, send me the file and I will test the changes. Thanks!



Can you write Arduino C code? Try adding some print statements to see how Stopped and Stopping are changing.

Here's a new version.



Fine!!! Works!!! I am writing Arduino C code and saw an error in the previous version
Stopping != Stopping; and tried to change to
Stopping != Stopped;, but didn't think to swap the lines Stopped = false; for the button trigger to work properly.
Thank you Peter for the revision!



The STOP button should work in all three modes (did you try it?) but I suppose it's really only useful in the Large Display mode.

It's annoying it needs another button. How about if there were an extra mode:

Large display
Large display Stopped
Small display
Poincaré display

It might be enough just to change line 58 into

```
enum TMode { mdLargeECG, mdLargeECGStopped, mdSmallECG, mdPoincare, mdBattery }  
mode = mdLargeECG;
```



Yes, STOP works in all modes. And I liked it. Each mode is important for conservation. For example, extrasystole can be fixed on a small display and shown to the doctor or photographed. Personally, my opinion is that everything turned out well and understandable. An extra button is not a problem.



Hi Peter!

I continue to test the program. Today I caught one moment in the STOP function. As long as the electrodes are well connected to the body, the STOP function works well, but if at the time of studying the recording one of the electrodes is accidentally disconnected, the noise level immediately increases, the display is reset and the screen is cleared. If you press the STOP button again (to turn off the STOP mode), the measurement will continue. I connect the electrode again and continue the measurements. Can turn off the reading of the receiver for the duration of the STOP mode, or somehow block the measurements in another way so that nothing interferes with the study of the saved graph on the screen?



I don't understand why a bigger signal would make a difference.

Is it to do with "Lead Off"?

Maybe not call CheckLeadsOff() if it's "stopped":

```
if (! Stopped)  
CheckLeadsOff();
```



I commented out the line: // CheckLeadsOff ();
Everything is good now! Thanks!



But it will not now report if the leads are off. I suppose it's obvious if RA or LA comes off but what about RL?



Yes, the RA and LA indicators no longer work, but this does not interfere with the measurement. When you disconnect any of the electrodes, the EKG deteriorates greatly, and this is clearly visible.



If I were to consider connecting this to a PC/USB while using the easy solution would be opto-isolators on the USB data lines and no power pass-thru, continue to power device via battery. As for Data logging, a day is 17Mbytes but what if it were compressed rather than raw? Perhaps a good FRAM chip setup would work.



> If I were to consider connecting this to a PC/USB while using the easy solution would be opto-isolators on the USB data lines and no power pass-thru, continue to power device via battery.

Personally, I would be happy to use such a device during development but you will have to make up your own mind.

> As for Data logging, a day is 17Mbytes but what if it were compressed rather than raw?

I've not tried compression on an Arduino. The examples I've seen concern storing large amounts of compressed text in ROM so the Arduino only has to be able to de-compress the data.

You could google for
arduino lz compression
arduino Huffman coding

I find it hard to imagine that LZ or Huffman would work well. IIRC, LZ requires that you remember the last thousand (?) samples and encode the latest data by referring back to similar data in the past.

I think Huffman needs the whole data file at the beginning so it can build its tree.

If you google
ECG compression
you get lots of papers. People are getting between 4:1 and 20:1 compression ratios. But the best seem to use wavelet transforms. I doubt if an Arduino can do that much maths fast enough.

My guess is that you'd be better to start with something simple like delta encoding; google for
arduino delta encoding

Most "delta" should be small (the difference between one sample and the next) and you can encode each delta in just a few bits.

Peter



Thanks for a great project, Peter!



Thanks. Your build looks good. Are you going to put it in a box?



Yes of course. When I complete the assembly, I will show the photo.



Thanks for a great project, Peter!
We are looking forward to your further developments.

