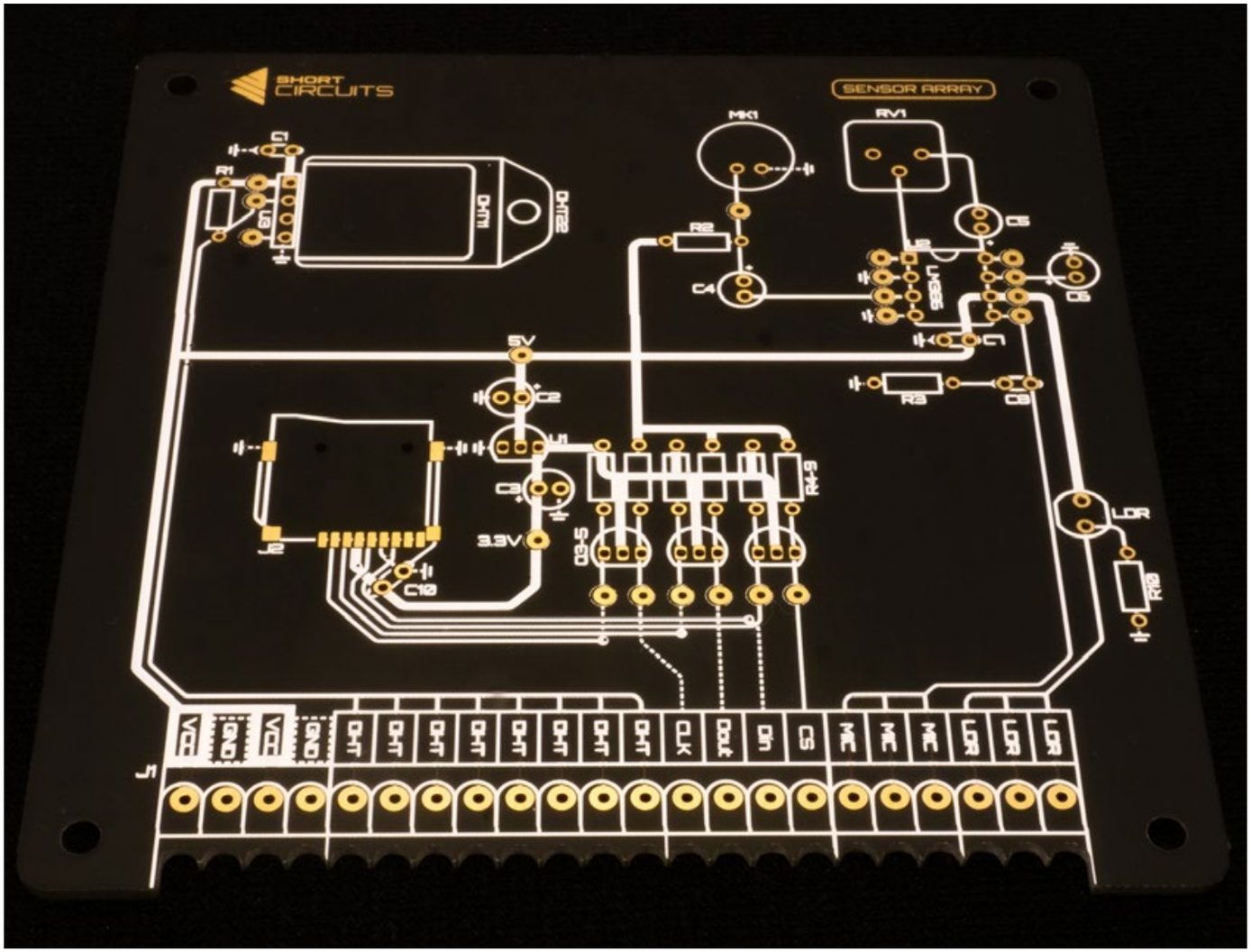
SHORT CIRCUITS




# SENSOR ARRAY

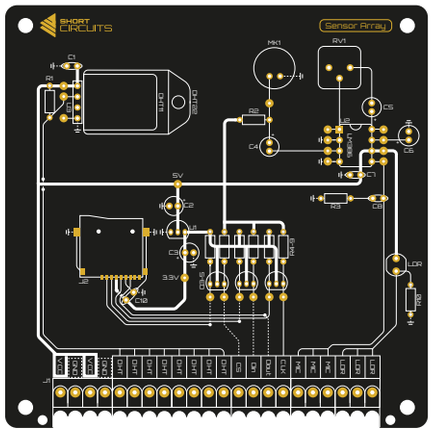
## INSTRUCTION MANUAL

# SENSOR ARRAY

The SENSOR ARRAY is just that, an array of sensors. The Motherboard and other Arduino based microcontroller boards use various inputs and outputs to form functions for a device that is of some use. The Sensor Array gives the Motherboard access to data about the surrounding environment. This data can be saved for later, or used to trigger other actions like outputting to a display, or triggering an alarm.

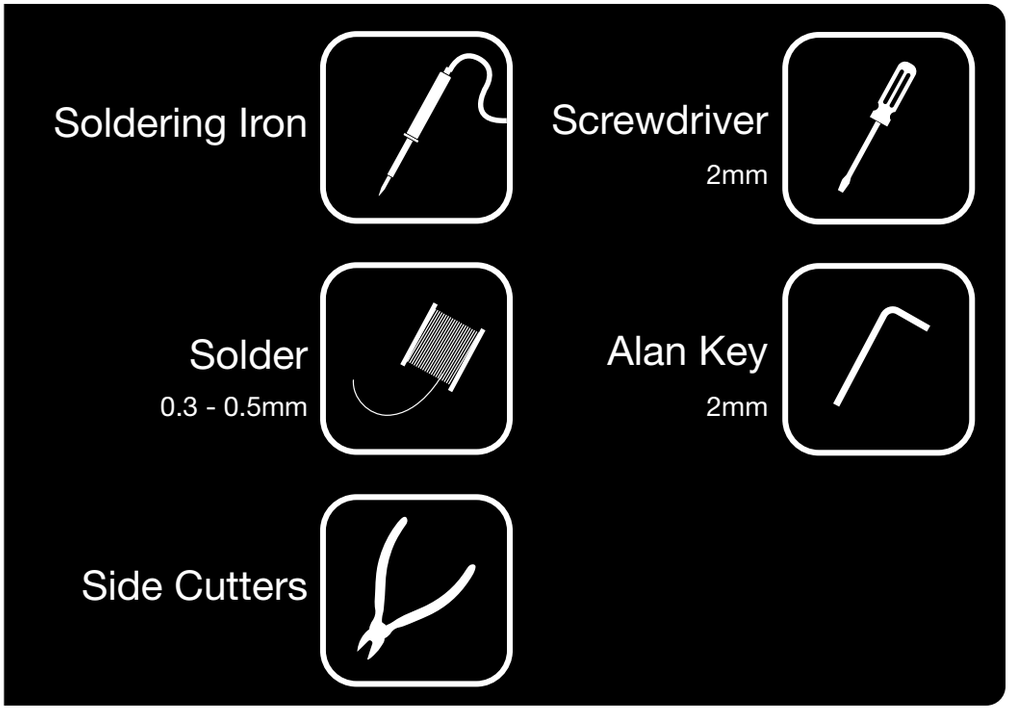
The sensors available on the Sensor Array are temperature and relative humidity via the DHT11, sound via a microphone and amplifier circuit, and light via a light dependent resistor. We have also included a Micro SD card slot and a logic converter circuit to convert our microcontroller's 5V logic to the 3.3V needed for the SD card's logic.

# Kit Contents

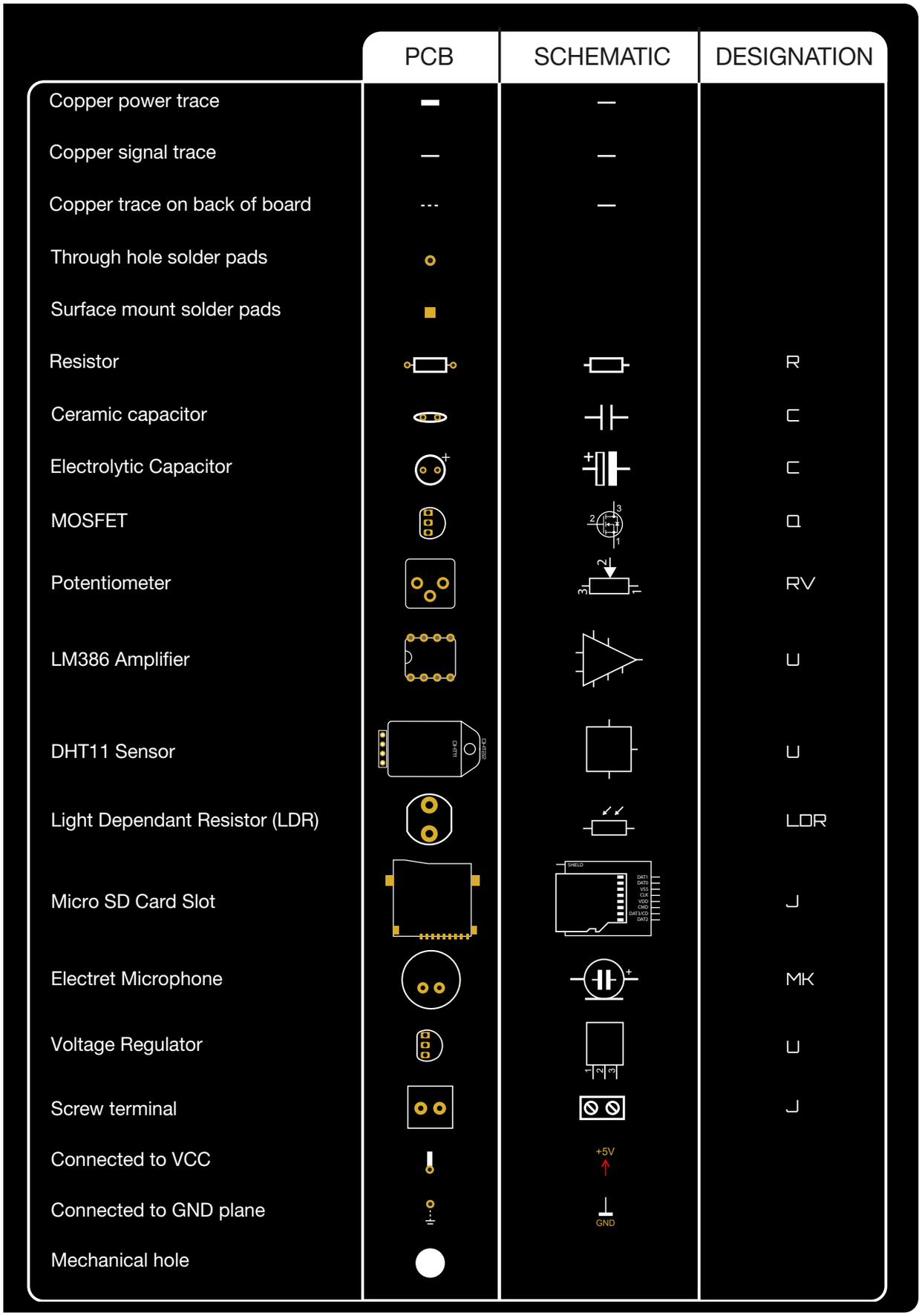


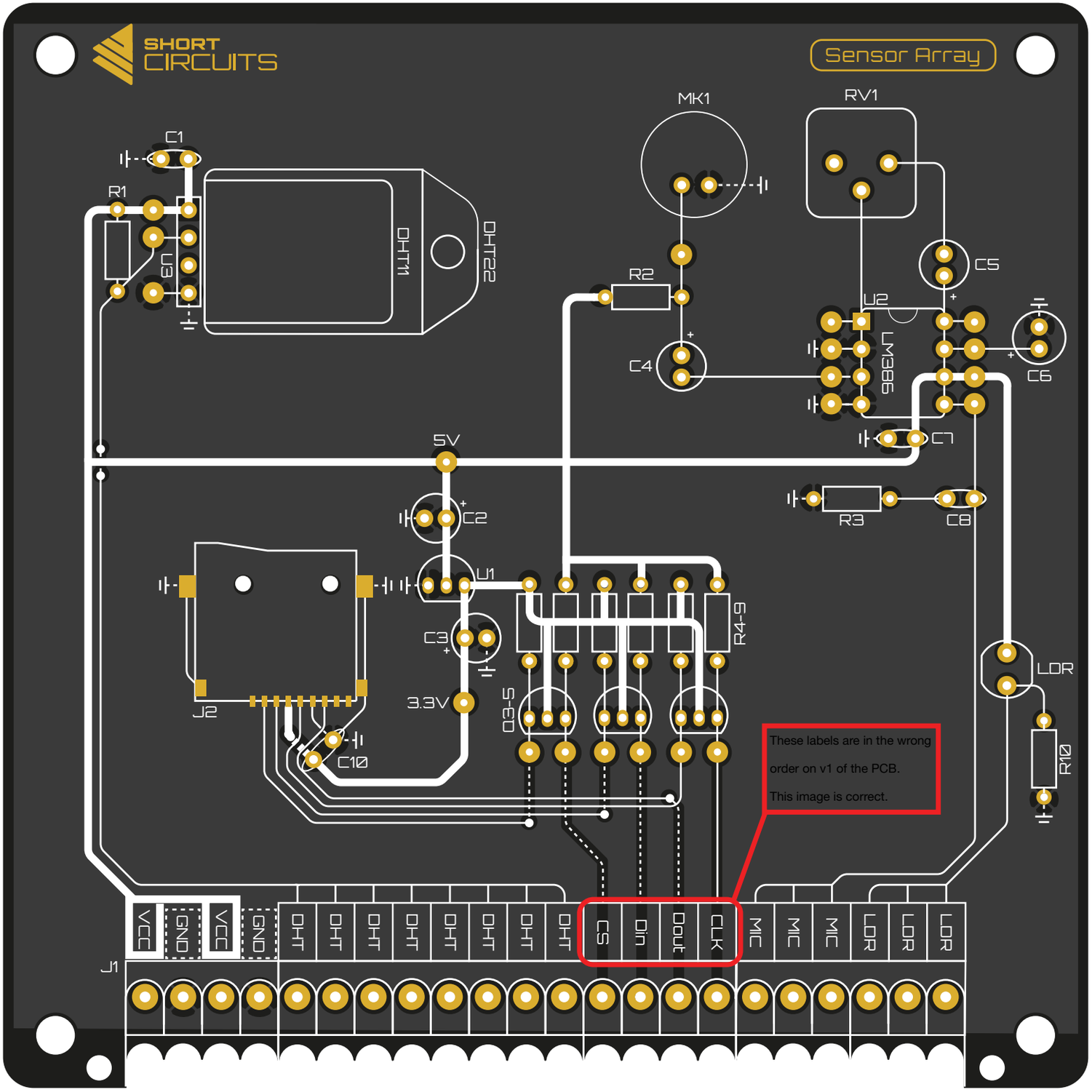
1 x Printed Circuit Board

- 1 x DHT11 Sensor
- 1 x Micro SD Card Slot
- 1 x Microphone
- 1 x Light Dependant Resistor
- 1 x LM386 Amp
- 1 x DIP8 Socket
- 3 x 100nF Ceramic Capacitors
- 1 x 47nF Ceramic Capacitor
- 5 x 10uF Electrolytic Capacitors
- 1 x 3.3V Regulator HT7533
- 3 x 2N7000 MOSFETs
- 9 x 10K Resistors
- 1 x 10 Ohm Resistor
- 1 x 10K Potentiometer
- 11 x Screw Terminals
- 4 x 5mm M3 Hex Screws
- 4 x Female/Male Standoffs



# Circuit - Symbols and Designations

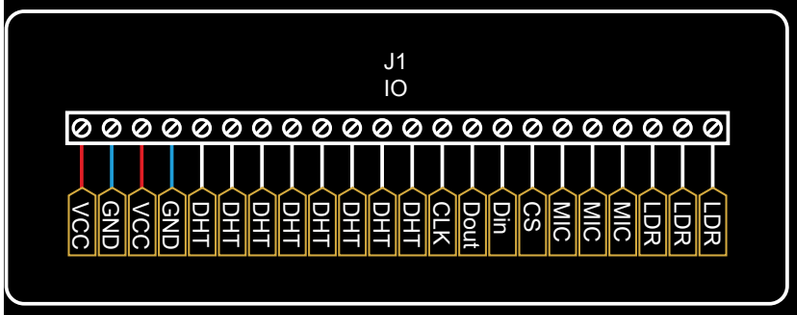




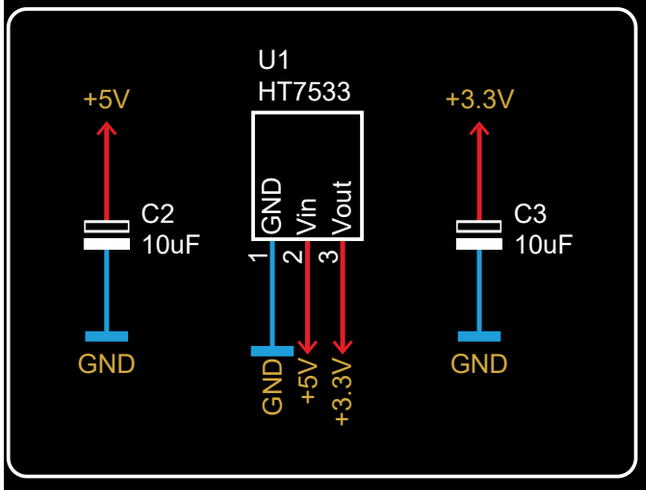

Ground (GND) Copper Area On Back of PCB

# Circuit - Schematic

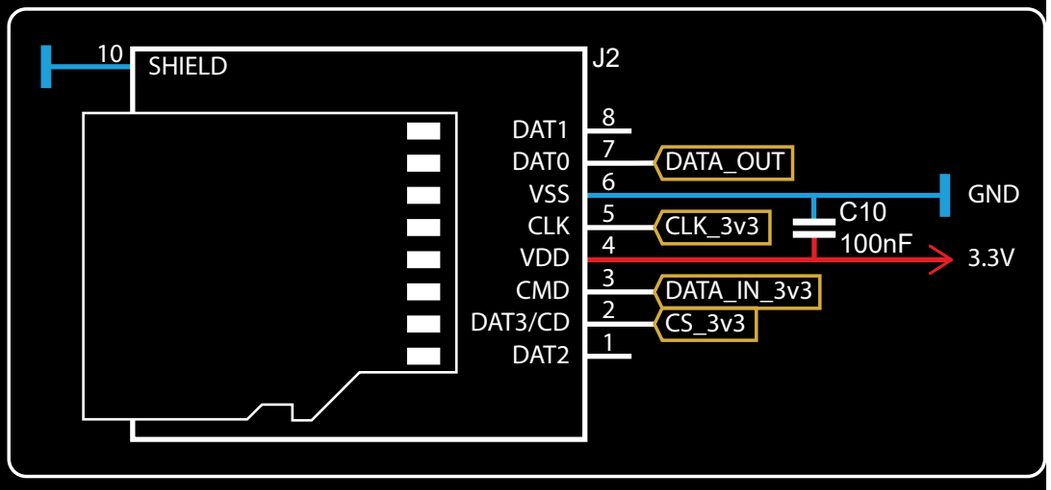
## Terminal Block IO



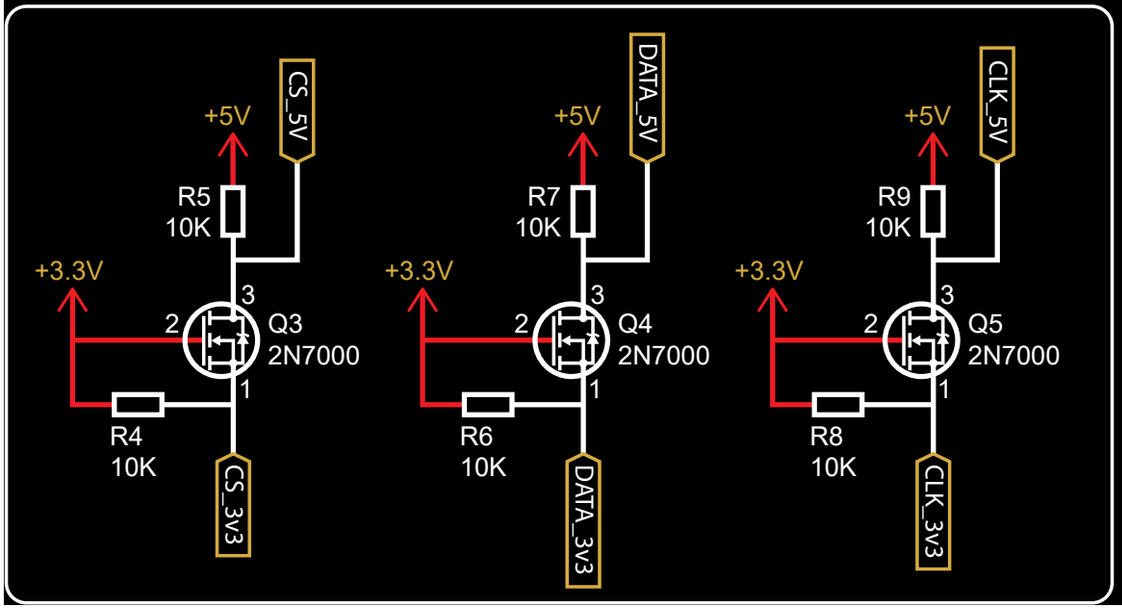

## 3.3V Voltage Regulator



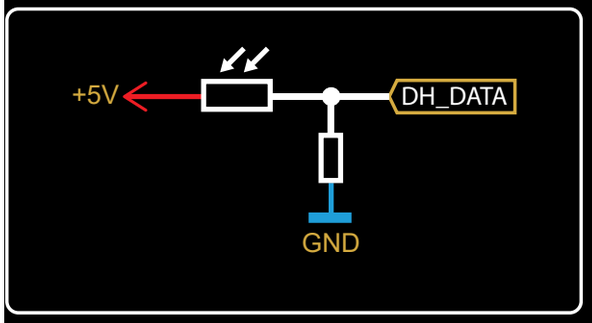

## Micro SD Card Slot



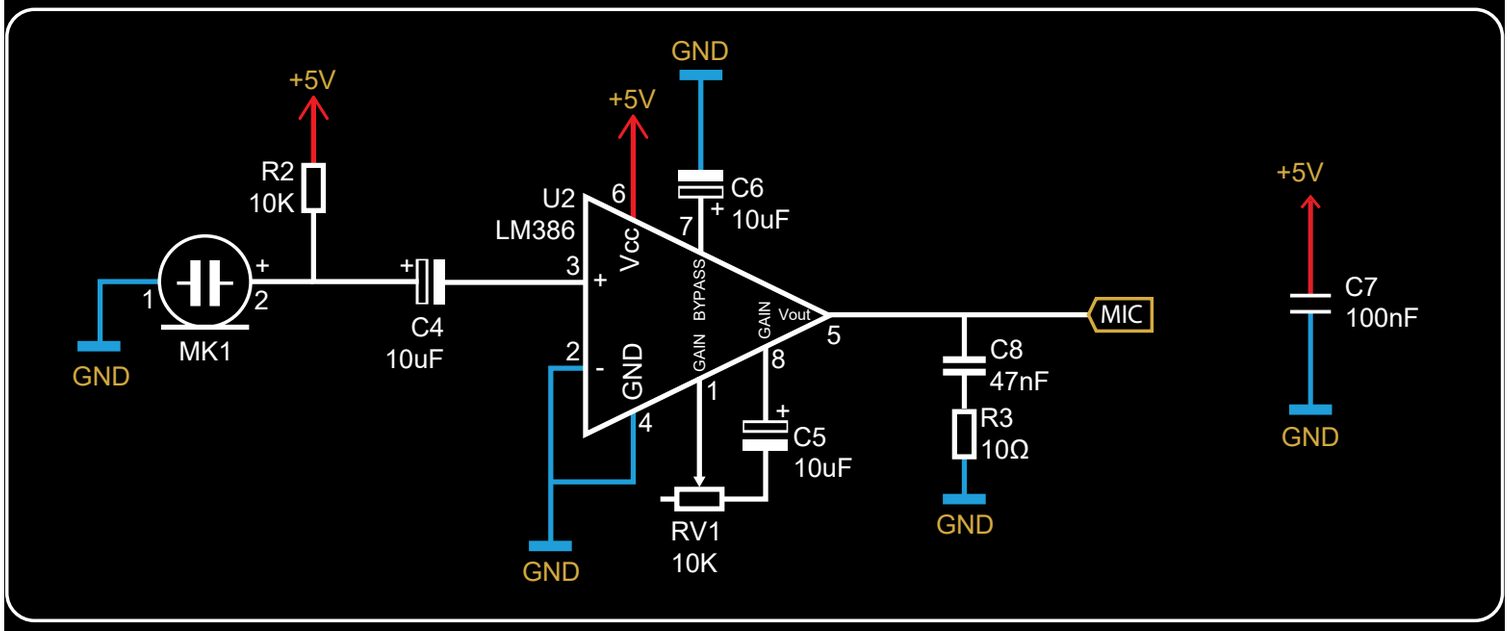

## 3.3V Logic Level Converters



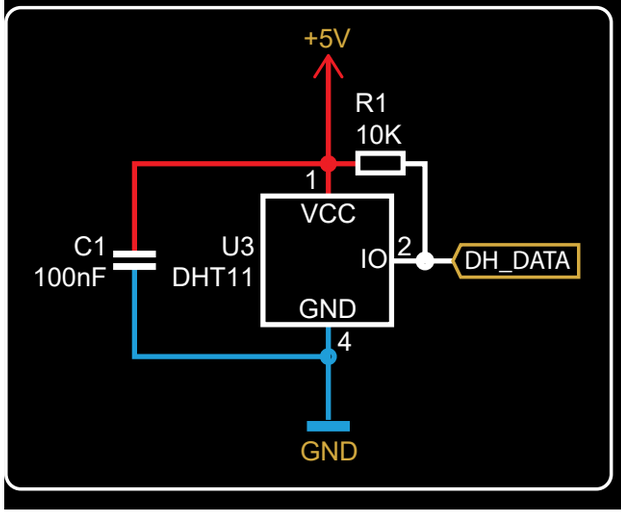
## Light Dependant Resistor (LDR)




## Microphone and Amplifier Circuit




## DHT11 Sensor



# Bill of Materials (BOM)

| Designation | Value | Name | Footprint / Pitch | Datasheet |
|---|---|---|---|---|
| C1, C7, C10 | 100nF | Ceramic Capacitor | 2.54mm |  |
| C2 - C6 | 10uF | Electrolytic Capacitor | 2.54mm |  |
| C8 | 47nF | Ceramic Capacitor | 2.54mm |  |
| J1 | 11 x 2 pos | Screw Terminal | 3.5mm |  |
| J2 | Micro SD | Micro SD Card Slot | |  |
| Q3-5 | 2N7000 | N Channel MOSFET | TO-92 |  |
| R1, R2, R4 - R10 | 10K | Resistor THT | 7mm |  |
| R3 | 10Ω | Resistor THT | 7mm |  |
| RV1 | 10K | Resistor THT | |  |
| LDR | 5K - 0.5M | Photoresistor THT | 3mm |  |
| MK1 | -52dB | Electret Microphone | 2.54mm |  |
| U1 | HT7533 | 3.3V Regulator | TO-92 |  |
| U2 | LM386 | Amplifier IC | DIP8 |  |
| U3 | DHT11 | °C/RH Sensor | |  |

**Micro SD Card Slot (J2)**

SD Cards can be used to store information from sensors or interactions over long periods of time. You could use it to log the temperature of a room over the course of a day, or even a year. The data could then be imported into a spreadsheet and graphs can be made! Marvelous. This is just one example, but there are many more.

To interface an SD card with a microcontroller running Arduino code, we need to use SPI mode. SD cards can interface using a more complex protocol called SDIO. This is what mobile phones and cameras use, but it is far too complex for a microcontroller to manage.

The labels shown on the datasheet and those on the SD slot's diagram below are SDIO pins. They are there for your reference as datasheets often only mention these. SPI uses 4 connections; a Chip Select, 2 data lines, and a Clock Pulse. We have labeled these from the SD slot's point of view (the "slave" in the master slave analogy. See the Motherboard Manual p29 for more info).

PLEASE NOTE: The labels on v1 of the PCB are wrongly labelled. Please refer to the diagram on page 5 for the correct labels.

**3.3V Voltage Regulator (U1)**

SD cards run off 3.3V and so does their logic. Our system runs on 5V, so we use a 3.3V Voltage Regulator to provide the 3.3V to the card and to the logic level converter circuitry. We have used a 7533 for this as it provides the 3.3V we need at a max current rating that exceeds our requirements.

**Logic Level Shifting (Q3-5, R4-9)**

Logic level shifting is needed when two communicating devices read a High (digital 1) at different voltages.

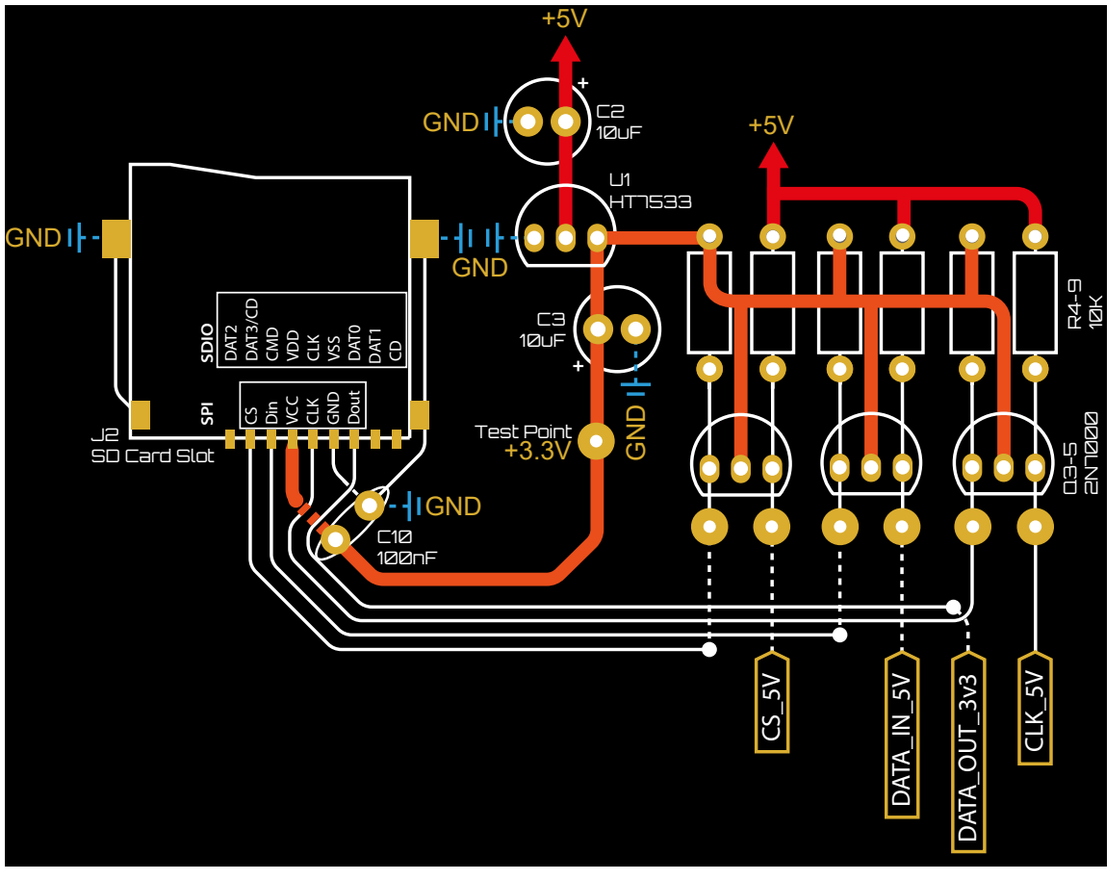
The SD card regards 3.3V as high, and the Microcontroller sees anything from 3-5V as high.

This is fine when the SD card sends a 1 to the microcontroller, as 3.3V is still considered High and won't destroy anything. This is why DATA_OUT is not converted in our circuit. The problem arises when the microcontroller sends a 1 (5V) to the SD card. As the card is only rated for 3.3V, this can cause damage.

There are a few ways to achieve logic level shifting. One method is to use two resistors to create a voltage divider (see Resistors in the component index). However, this method can be somewhat unstable and only works in one direction.

Another method uses N-Channel MOSFETs with some pull-up resistors. This method is stable, fast and works in both directions. SD cards in SPI mode are limited in speed so the 2N7000 works fine. For faster switching use a dedicated logic level MOSFET.

Arduino, and therefore the Motherboard will only read SD or SDHC Micro SD cards up to 32GB. These must be formatted to FAT32. This can be done by right clicking the SD card in an explorer window on a Windows PC, choosing format, selecting FAT32, then formatting.



# Circuit - Light Dependant Resistor (LDR)

**Light Dependant Resistor**

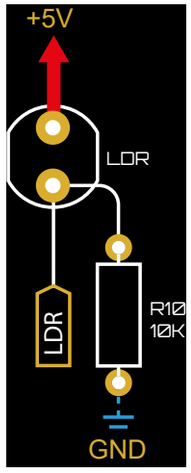
A light dependent resistor is a simple device that changes resistance depending on how much light is hitting it. Increasing the light intensity decreases the resistance.

**Resistor (R10)**

Because the LDR simply acts as a resistor with varying resistance, connecting it between 5V and the analog pin of the microcontroller won't work. This would just vary the current that can pass through the resistor, but the voltage would stay pretty much the same.

To achieve the effect we need for an accurate reading on an analog pin, (a value between 0V and 5V), we need to use another resistor to form a voltage divider. The LDR we are using has a resistance range somewhere between ~100Ω and ~0.5MΩ (500K ohms).

If the LDR has a lot of light shining on it, and has a resistance value of around 100Ω, the analog pin would read closer to 5V (as this is the path of least resistance) and show an analogRead value of close to 1023.



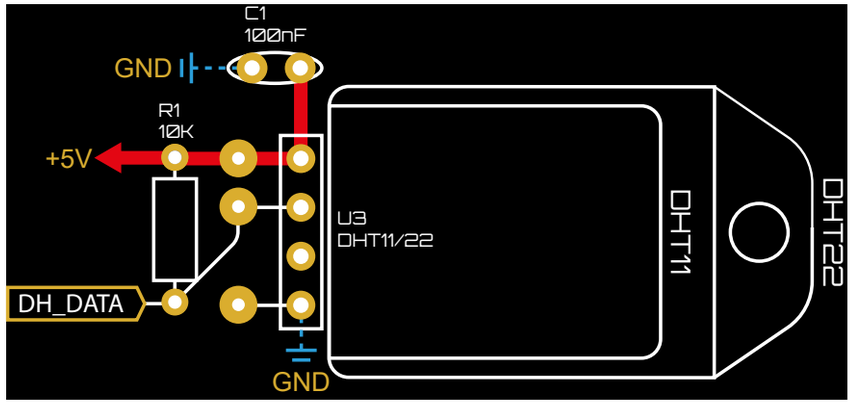

If the LDR was in complete darkness and had a resistance value of 500K, the path of least resistance would be towards GND. This would result in a value close to 0V and an analogRead value close to 0.

If there was just enough light for the LDR to have a resistance of 10K, we would read a value of 2.5V and an analogRead of ~512.

These values can then be used to control other functions. Maybe we would like to turn on an LED when the light level gets too low, or we could map the value to a range that we can more easily understand, like 0-100, then display it on a digital display as a percentage.

**DHT11 Sensor (U3)**

The DHT11 is an inexpensive sensor that detects Temperature and Relative Humidity. It is a single data line device with some complicated timing to deal with in the code. Fortunately there are very useful libraries available ("DHT Sensor Library" by Adafruit for example) to handle all the complex stuff, so you can concentrate on the data itself.




One of the limitations of the DHT sensors is that they tend to interrupt anything else that the microcontroller is controlling while they are transferring data (a flicker of the display while the temperature is being read for example).

The DHT11 has a more expensive cousin, the DHT22. This sensor has a better temperature and relative humidity range than the DHT11, but this is only useful in certain situations. They are interchangeable though, so if you find yourself needing more range, then swap the DHT11 out for a DHT22.

**Additional Components**

As with any sensitive electronic device, a bypass capacitor is always useful to smooth out any noise. Also, the DHT11's data line needs to be held high according to its datasheet. So a 10K resistor is added between it and Vcc.

# Circuit - Microphone & Amplifier

## Sound Sensor

This microphone and accompanying amplifier circuit are used as a sound sensor rather than an audio recording device. We can use it to listen for a clap which can trigger another action. We can also use it to measure relative ambient/background noise levels. With a bit of calibration and testing this could be used to warn you if you're being too loud!

## Electret Microphone (MK1)

An electret microphone (a type of condenser mic) acts as a capacitor. It has two conductors separated by an insulating layer (air in this case). When sound waves hit the first conductor (the membrane) it vibrates. This changes the distance between the membrane and the other conductor. This acts as a variable capacitor that changes depending on the sound waves hitting it.

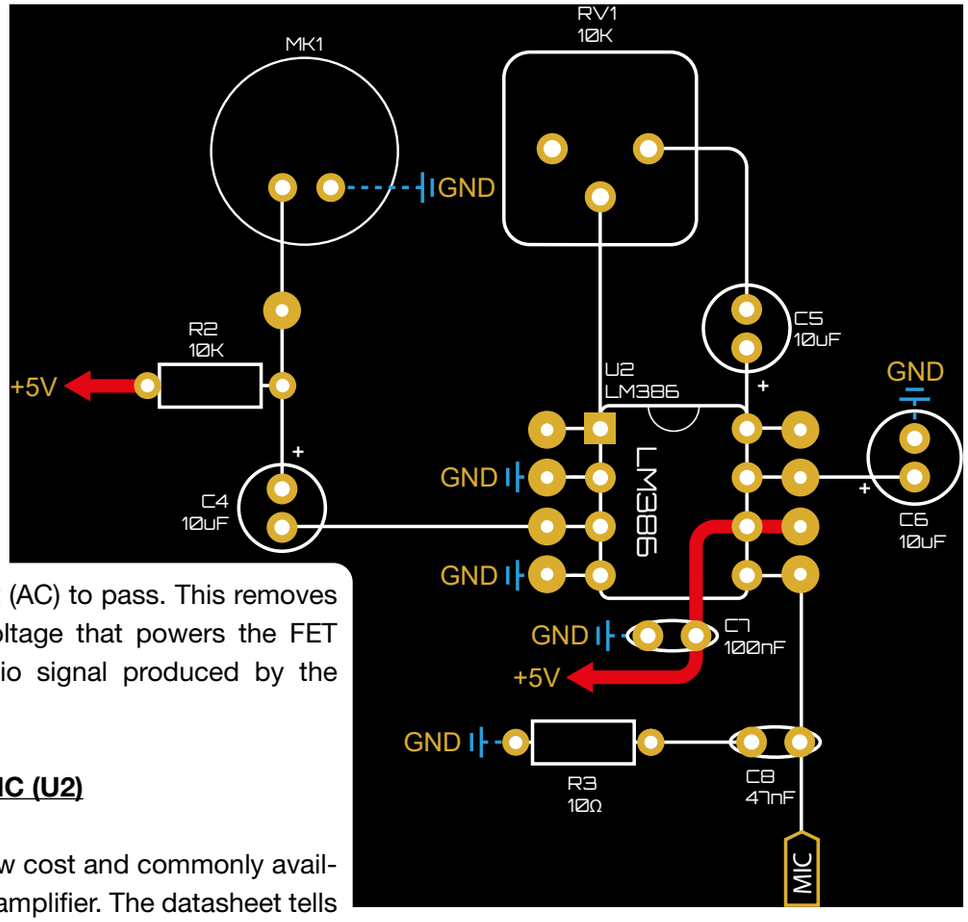
The word "electret" is formed from the words "electrical" and "magnet". This is because the membrane is permanently electrically charged, which makes it act like a magnet. This helps convert the capacitance to voltage, which is then amplified inside the microphone by a transistor.

## Load Resistor (R2)

The built in Field Effect Transistor (FET) needs to be powered for it to amplify the voltage generated by the microphone. This is where the load resistor comes in. This resistor, tied to Vcc, will provide a small amount of bias for the transistor to operate.

## Coupling Capacitor (C4)

An electrolytic capacitor is used in series with the output of the microphone. A capacitor in series will block direct current (DC) and allow alternating current (AC) to pass. This removes the DC supply voltage that powers the FET from the AC audio signal produced by the microphone.




## LM386 Amplifier IC (U2)

The LM386 is a low cost and commonly available audio power amplifier. The datasheet tells us that the amplifier needs minimal external components, has an operating current drain of 4mA, and offers voltage gains from 20 to 200. The datasheet also provides a range of example circuits that can help get you started. We have used the minimum component example with a few changes.

## Bypass Capacitor (C6, C7)

Both of these capacitors help filter out any noise from the power supply that would otherwise enter the amplifier.

## Output Filtering (C8, R3)

C8 and R3 are used as a filter. This type of filter is called a Boucherot Cell and is used to dampen high frequency oscillations. It provides an escape path for high frequency oscillations to prevent them building up.

## Gain Control (RV1, C5)

Normally you would control the volume using a pot attached to the input of the amp. This would allow you to lower the volume from its max (It doesn't add volume). To enable us to pick up background sounds and get an idea of general ambient noise we added the pot to the gain control rather than the input. This enables us to change the amount of gain (increase in volume from input to output).

The datasheet shows us an example of a static resistor in series with a capacitor between pins 1 and 8. Different values of resistor will give us a different gain. Without a resistor (0Ω), the gain is set to 200 (so when the pot is turned fully to the right). When the pot sits at around 2KΩ the gain is set to 20 (yes we should have used a 2K pot...). You can experiment with the code and the pot to get the desired effect.

# Assembly Instructions

**General Soldering tips**

1. ALWAYS KEEP YOUR TIP CLEAN

To ensure the soldering iron can transfer enough heat from it to your solder/component leg you must keep the tip clean and shiny. A dull tip means the outside layer of metal has oxidise. This oxidised layer is a poor transferrer of heat. Because of this, you will have to hold the tip against the component for a longer period of time, which can result in the component failing. It's also very frustrating.

To keep your soldering iron tip clean, wipe it on a wet sponge or wire ball, then apply some solder to coat it, then wipe it again. Ideally, you should do this after every component. At the very least, do it after every 4 components.

2. CONTACT

When soldering, make sure the tip of your iron is making contact with both the leg of the component and the pad on the PCB. Apply heat to the area, then, within a second or two, apply the solder to the point of contact.

3. HEAT

It is better to be too hot than too cold. As mentioned earlier, when the iron tip isn't hot enough, you have to hold it on longer. This allows heat to transfer into the component and could cause a failure. It is better to set your soldering iron a little hot so the solder melts instantly and flows around the leg of the component with ease. You can start at around 350ºC and adjust from there.

4. SOLDER

Leaded solder is much easier to work with, which makes it easier to learn with. It can be hard to find in some countries, but can often be ordered from China. There are potential health risks, but these are very low. Make sure you have a fan pointing away from your work area to blow the fumes away. Work in a well ventilated area.

Thinner is better. Working with a thick wire of solder can get messy. Use 0.4-0.5mm solder for more c ontrol. You will have to feed more into the solder joint, but you have more control when there are other pins close by that you want to avoid. This is essential when soldering surface mount components and Integrated Circuits.

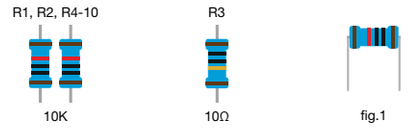
5. SAFETY

350ºC is obviously very hot. Stuff catches fire at this temperature. Skin fries. Please be careful. Remember to turn it off when you are finished. This will prevent a potential house fire and also save your iron tip from continued oxidation.

6. ANGLE OF ATTACK

When soldering, make sure you position the board so that it is easy to access the area you are working on. It is easy to make a mistake when you are trying to maneuver your iron into position around some obstacle. You may have to spin the board 180 degrees, or make sure you have snipped the legs off the previous component. It's easy to be impatient so try to plan ahead.

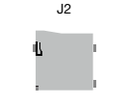
**Resistors**




Bend each leg of the resistor by 90 degrees, as close to the resistor as possible (fig.1). Then insert the legs into the correct holes by finding the correct reference (R1, R2 etc) on the board, or by using the picture on the next page as a guide. Polarity is not important with resistors, so either way is fine.

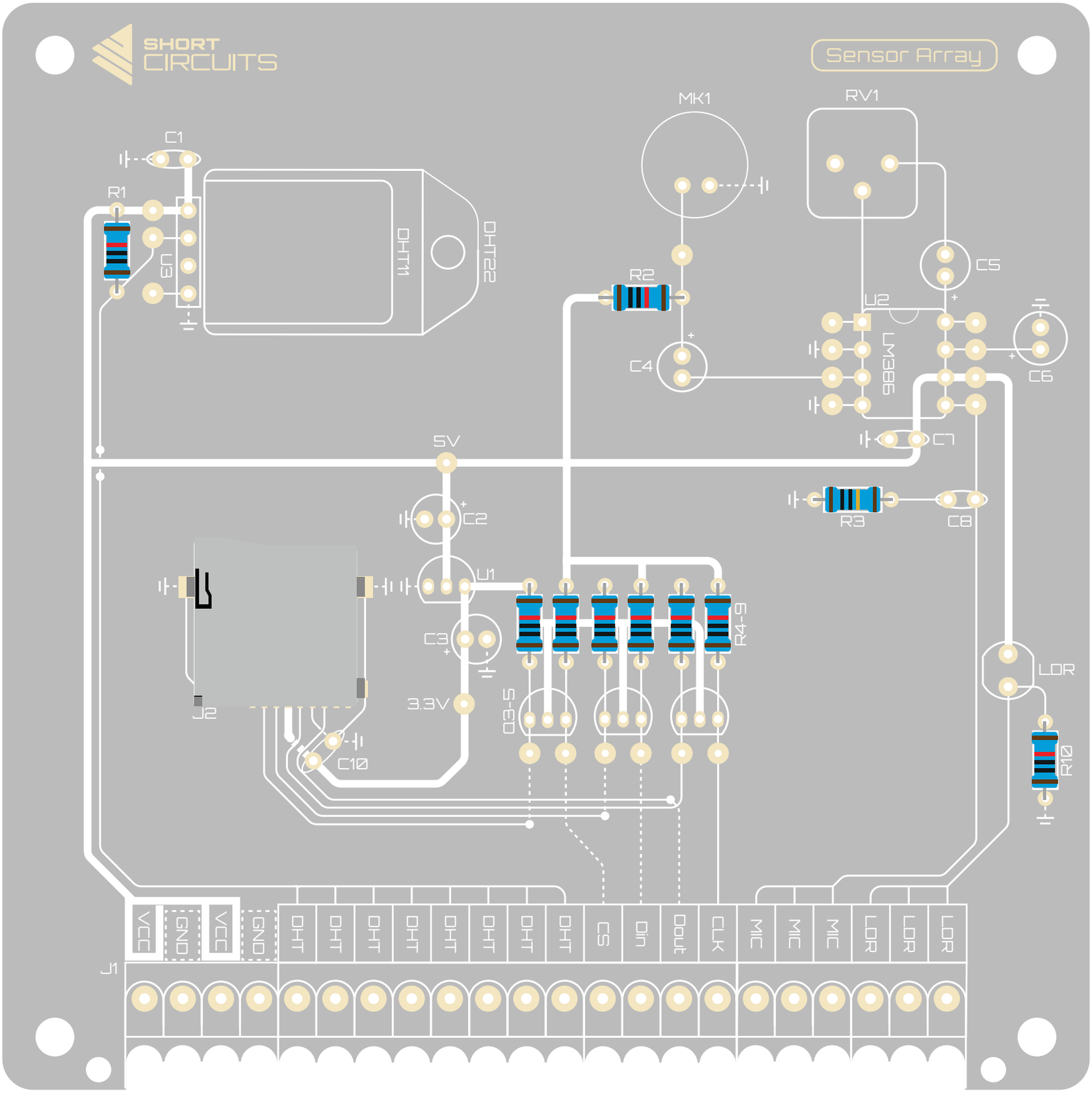
Hold the resistor in place with sticky tack or tape, then flip the board over, ready for soldering. Solder each leg by holding the soldering iron tip to the leg and the pad, then apply the solder to the joint. Let it flow around the leg and create a shiny cone. Use flush cutters to cut the legs off, then briefly hold the soldering iron to the joint to re-flow the solder.

**Micro SD Card Slot**




This is by far the most difficult component to solder in this kit. It is similar to the USB socket in the Motherboard kit, but slightly easier. If you would like to leave this to the end, make sure you leave C10 to the end.

Start off by placing the SD card slot in the correct position according to the footprint J2. While making sure the row of pins are lined up with the pads on the board, use


SHORT CIRCUITS
Sensor Array
MK1
RV1
C1
R1
DHT11
DHT22
R2
C5
U2
LM386
C4
C6
C7
5V
R3
C8
C2
U1
C3
R4-9
LDR
3.3V
J2
C10
R10
S-D3
J1
VCC
GND
VCC
GND
DHT
DHT
DHT
DHT
DHT
DHT
DHT
DHT
CS
Din
Dout
CLK
MIC
MIC
MIC
LDR
LDR
LDR

ASSEMBLY

sticky tack to hold the component in place and solder the four pads around the sides.

Apply some flux to the pins to help prevent bridges. Now clean your tip and apply a little solder to it. Press the tip down on the first pin in the row and drag away from the pin. This should leave enough solder to bond the pin to the pad. Try to avoid getting any solder on the second pin. A thin tip is good for this. Now repeat the process for each pin.

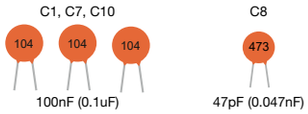
If you accidentally bridged two or more pins, apply more flux to the area, clean your tip, then drag the tip between the two pins then wipe it clean. Repeat this until you have removed the bridge. If it isn't working, add more flux. If the solder isn't liquefying enough increase the temperature of the iron a little. If this doesn't help, make sure your tip is shiny. If it isn't, you may need to clean it more thoroughly and add solder to make it shiny. If the tip dulls quickly, or the solder refuses to stick to it, your iron may be too hot.

A video guide will be made shortly after the manual is published.

## Light Dependant Resistor (LDR)


Solder this like any other two legged through hole component. Add some tack, flip the board and solder the joints. This component isn't polarity sensitive.

## Ceramic Capacitors




Install the ceramic capacitors into the corresponding holes (C1, C7, C8, C10). They will easily fall out, so add some sticky tack or tape to secure them. Polarity does not matter. Flip the board and solder the pins to the pads, then trim and re-flow.

## IC Socket (DIP-8)

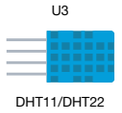

We are going to solder a socket to the PCB rather than the chip itself. We do this to avoid heating up the IC and to enable us to swap it out if it stops working.

Insert the socket into the area marked U2. Pay attention to the semi-circle indicator and match the one on the socket with the one on the board.

Now, stick it in place and solder 1 corner pin, then the diagonally opposite corner. Now check alignment and whether its flush with the board. If one corner is too high, add pressure to it while heating the pin underneath.

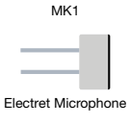
Now you can solder all the other pins knowing that the socket will stay exactly where it needs to be. If you accidentally bridge two of the pins, wipe your clean iron tip through the center of them until they are separated.

## DHT11



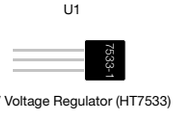

The DHT11 is a simple component to solder. The leads are generously spaced. Insert the leads into the corresponding footprint (U3). Orient the module so the holes in the case face upwards when the module is laying flat. Lay the module flat and fix it in place with some tack or a bit of glue. Now solder the pins to the pads. Snip and reflow. You can swap this out for a DHT22, available on the usual online retail websites.

## Microphone



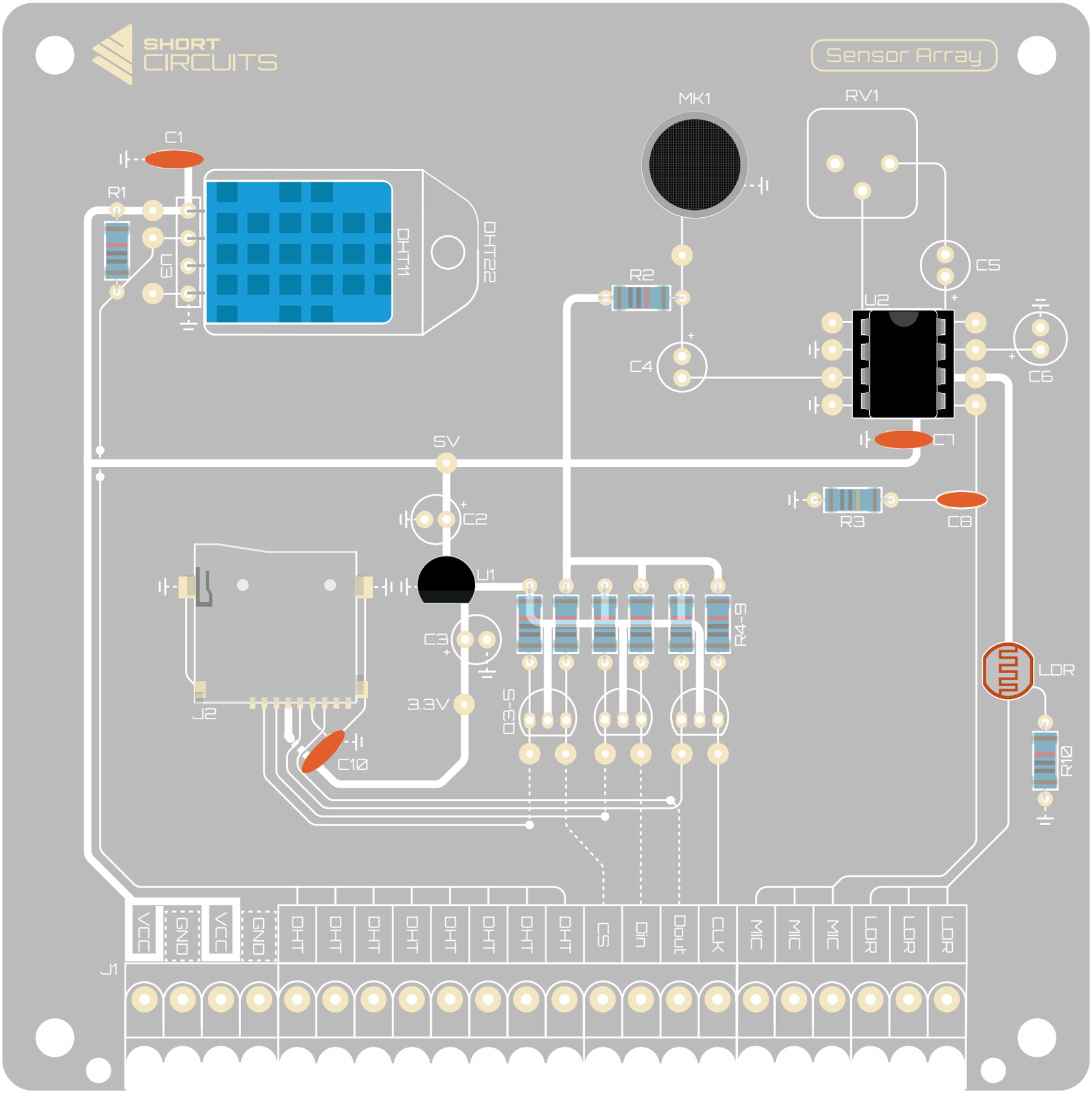

The microphone is polarity sensitive, so make sure the mic lines up with the footprint when inserted. Solder it like any other through hole component.

## 3.3V Voltage Regulator (HT7533)




The HT7533 is the first of two components that use the TO92 form factor. TO92s can be tricky to solder because the legs are often close together. We have spaced the pads out a little to make it easier. Notice the semi-circular shape of the TO92 looking from the top. Match this with the footprint on the PCB.

The outer legs will spread out a little as you

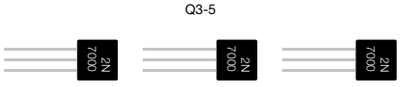

SHORT CIRCUITS
Sensor Array
MK1
RV1
C1
R1
DHT11
DHT22
C5
R2
U2
C4
C6
C7
5V
C2
R3
C8
U1
R4-9
C3
LDR
J2
3.3V
S-03
C10
R10
J1
VCC
GND
VCC
GND
DHT
DHT
DHT
DHT
DHT
DHT
DHT
DHT
CS
Din
Dout
CLK
MIC
MIC
MIC
LDR
LDR
LDR

ASSEMBLY

insert it into the holes. Don't push it too far as the legs will spread too much and crack the black casing. Stop when there's a 2-3mm gap. Add sticky tack to hold in place, flip, solder, snip and re-flow.

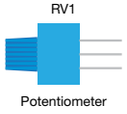
If you accidentally bridge two pads together, clean your iron, re-tin the tip, then drag the tip through the gap between the pads. This should separate them. If it doesn't, repeat the process. If your tip isn't clean, the solder won't stick to the tip when you drag it through, so make sure that tip is clean and re-tinned.

**MOSFETs (2N7000)**



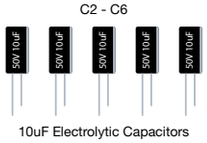

The 2N7000 MOSFETs are also in TO92 packages, so follow the same instructions as the HT7533.

**Potentiometer**




Insert the legs of the Potentiometer into RV1 on the board. Hold the potentiometer with some sticky tack and align the blue case with the footprint. Flip the board and solder the three pins. Snip and re-flow.

**Electrolytic Capacitors**

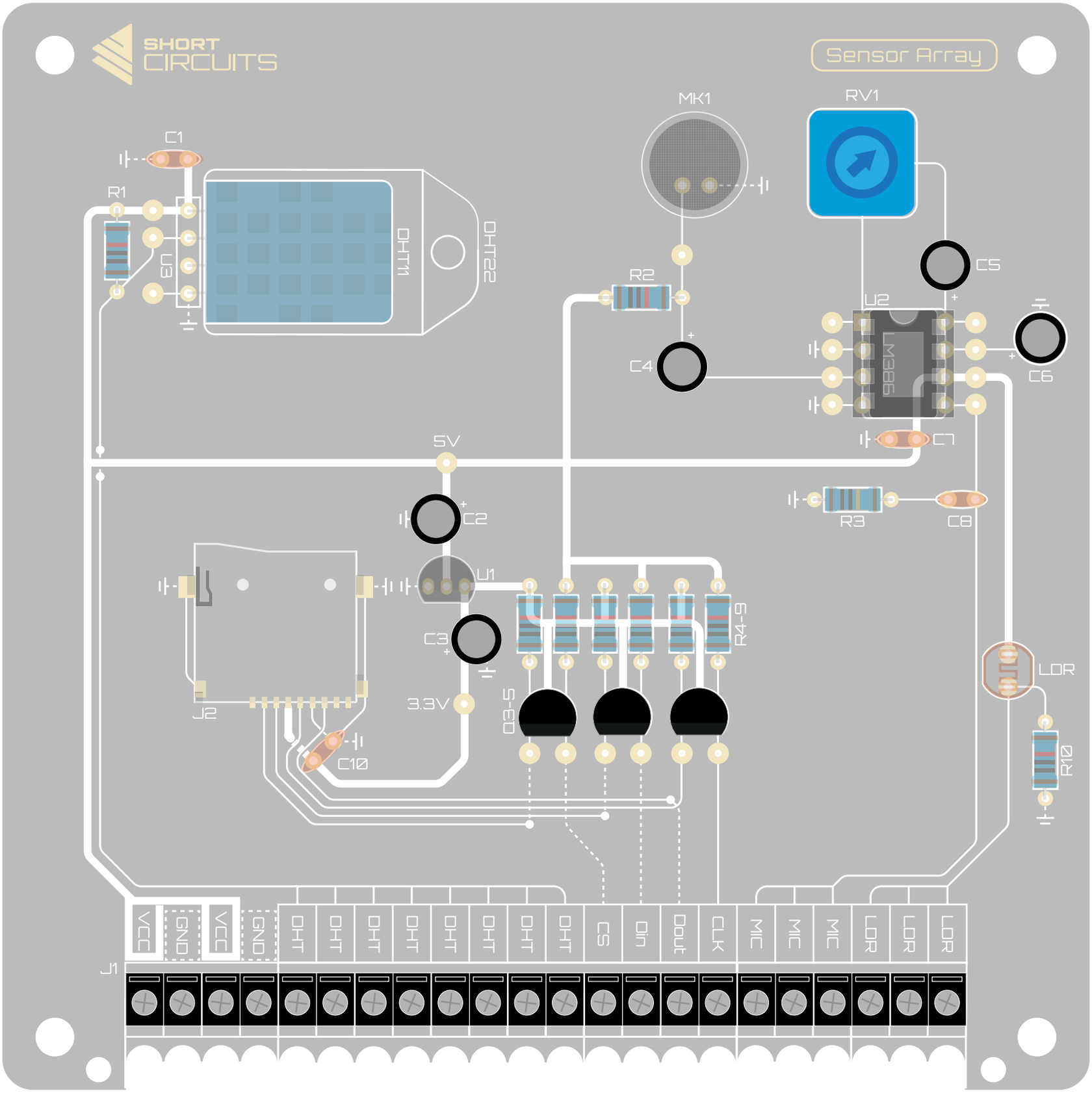



Electrolytic Caps can be soldered upright or laying against the PCB. This kit is designed to have them standing up. Other than being tricky to hold in place, the caps are easy to solder. Build up some sticky tack around the cap to hold it in place or use some tape. Then solder the pins as usual.

**Terminal Blocks**

The terminal blocks are interlocking, so slide each of them together to make a chain of 11. To make this line of terminal blocks fit in their holes, you may need to squeeze the blocks together a bit.

If you are using the Rack case to display your boards, I would suggest clipping the legs off. These can interfere with the wires and cause a short when they are sandwiched between the case and the board.


SHORT CIRCUITS
Sensor Array
MK1
RV1
C1
R1
DHT11
DHT22
U3
R2
C4
C5
U2
M386
C6
C7
5V
C2
R3
C8
U1
C3
R4-9
S-03
LDR
R10
J2
3.3V
C10
J1
VCC
GND
VCC
GND
DHT
DHT
DHT
DHT
DHT
DHT
DHT
DHT
CS
Din
Dout
CLK
MIC
MIC
MIC
LDR
LDR
LDR

ASSEMBLY

# Testing for faults

Before you power on the board, there are a few things we need to do.

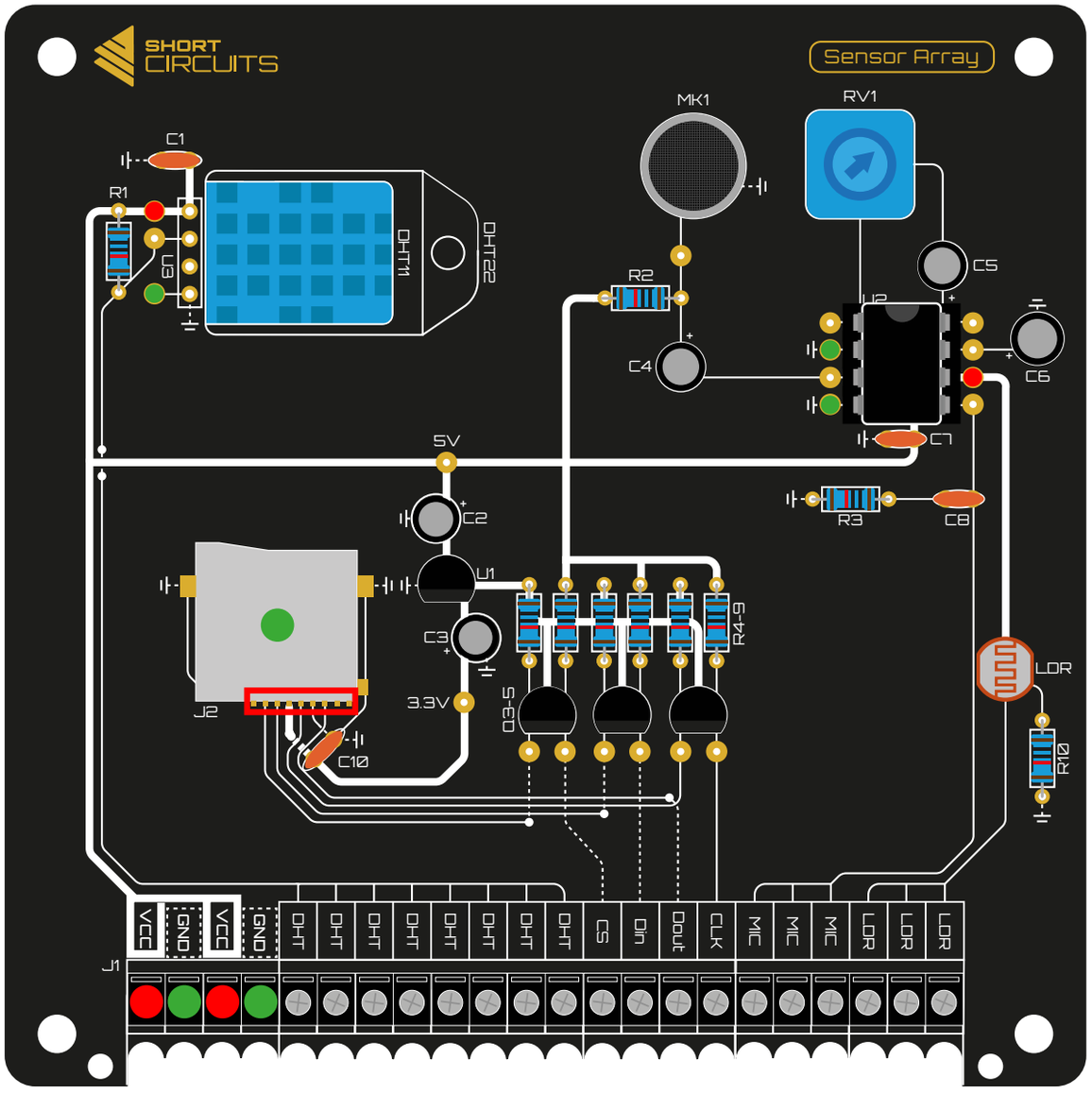
## Visual Check

Firstly we need to do a visual check of the back of the board. We are looking for solder bridges that connect two pads that aren't meant to be connected. A magnifying glass is a good tool to have when doing this. The most likely areas for this are highlighted with red boxes on the diagram of the back of the PCB and the SD card slot contacts on the front of the board.

If you see any solder bridges, bring your iron up to temperature and drag it between the two pads. You may have to repeat this a few times. Make sure your iron is clean or the solder won't cling to it. Add flux to aid the process.

## Short Circuits

Now we need to check for short circuits (the bad kind). If we have a short circuit somewhere on the board and we connect it to power, or a Motherboard kit, we could damage something. To check for shorts, get your multimeter and put it in continuity mode ( ) Make sure the black cable is plugged into the common socket and the red cable into the red socket that also has the Ω symbol on it. Touch the probes together and make sure it makes a sound. Now press the black probe on one of the green circles indicated in the diagram. These are all connected to the large ground plane on the underside of the board. Keep it held there while you press the red probe onto one of the red circles indicated in the diagram. The screw of the Vcc terminal and the metal case of the SD card slot are the most convenient. Making sure they are both making contact with metal, listen for a sound from your multimeter. If there is none, excellent, you don't have a short between Vcc and GND. You can jump over to the Power Test.



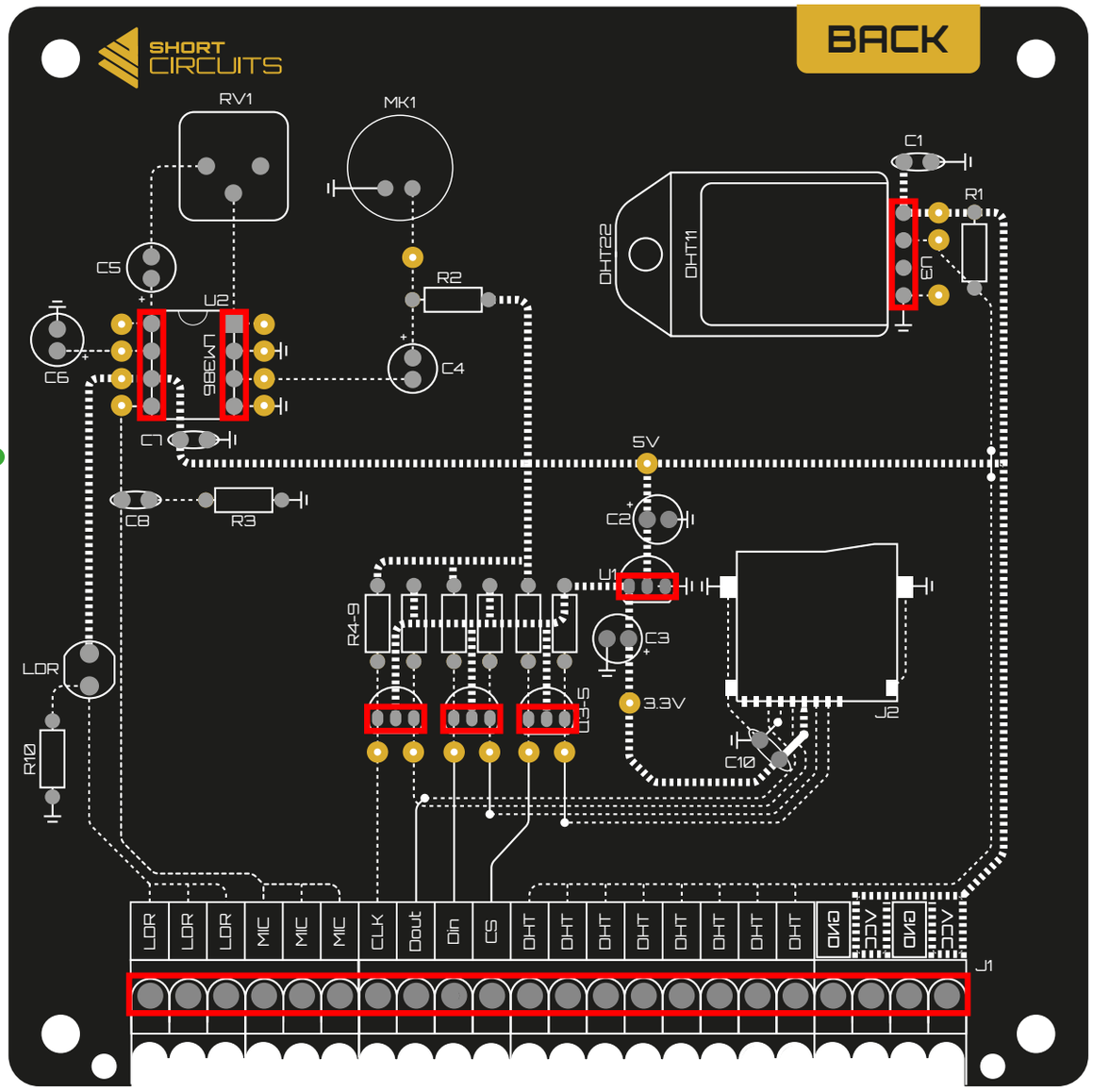

If you here a constant sound from your meter when the probes are in contact with Vcc and GND then you have a short. Check the back of the board again for solder bridges. Focus on the areas marked with boxes on the diagram on the next page. This is where Vcc and GND are closest.

## Polarity

Check all the components that are polarity dependent. In this circuit, that would be the DHT11, Electrolytic Caps, Microphone, Voltage Regulator, MOSFETs and the LM386.

## Powering the Board

We can now connect the Sensor Array to the Motherboard. Connect VCC, and GND to the Motherboards Power Out terminals. Check that the board is receiving 4.5-5V by




switching the multimeter to DC Voltage ( ) and putting the probes on the same points as last time.

Check the 3.3V regulator is working by placing the black probe on a GND point and the red probe on the test point labelled 3.3V, below the regulator (U1)

**Testing the Various Functions**

Now you connect each sensor to the Motherboard to test they work. Follow the “Data Logger” connection diagram in the Project Ideas section, then upload the test code (HERE). Open the Serial Monitor in the Arduino IDE and check the output. The text will tell you if your SD card is inserted and working, as well as 4 numerical outputs for temperature, relative humidity, sound and light. The temperature and humidity reading should be what you expect. The microphone reading will be all over the place. This is because the reading is a brief point on a rapidly oscillating AC audio signal. To better view this, try the test script which has been modified for use with the Serial Plotter (LINK). Upload this s ketch, then open the Serial Plotter which is found in Tools on the task bar (or press Ctrl+Shift+L). This will show you a graph of the data over time. Play around with the LDR by covering it or shining a light on it. Clap to see if the microphone reading responds. If something doesn’t work, go back and look at that part of the circuit. Check for soldering and orientation mistakes.

**Still having problems? Head over to the forums on our website for help and support. www.shortcircuits.cc**

# Coding Basics - DHT11

```
1  //Adding Libraries
2
3  #include <Adafruit_Sensor.h> // includes Adafruit sensor library
4  #include <DHT.h> // includes DHT library
5  #include <DHT_U.h> // includes DHT_U library
6
7  //Input Pin Variables
8
9  #define DHTPIN 9 // Digital pin connected to the DHT sensor
10 #define DHTTYPE DHT11 // Indicate the sensor type: DHT11 or DHT22
11 DHT_Unified dht(DHTPIN, DHTTYPE); // sets the pin and type
12
13 //Other Variables
14
15 int temp = 0; // variable to hold temperature data
16 int humid = 0; // variable to hold humidity data
17
18 //Void Setup - runs once at start
19
20 void setup()
21 {
22   Serial.begin(9600); // Initialise Serial Monitor
23   dht.begin(); // Initialise DHT Sensor
24 }
25
26 //Void Loop - repeats forever
27
28 void loop()
29 {
30   sensors_event_t event; // Prepares for the following events
31   dht.temperature().getEvent(&event); // Gets the temperature data from the DHT
32   temp = event.temperature; // Saves the temperature to a variable
33   dht.humidity().getEvent(&event); // Gets the relative humidity data from the DHT
34   humid = event.relative_humidity; // Saves the relative humidity to a variable
35
36   Serial.print("Temp: "); // print text to Serial Monitor
37   Serial.println(temp); // print variable's value and start a new line
38   Serial.print("R.Humidity: "); // print text
39   Serial.println(humid); // print variable's value and start a new line
40
41   delay(5000); // delay for 5 seconds
42 }
43
44
```

These code examples and explanations will test all of the Sensor Array's functions and teach you the basics. We will cover the following:

- Read the temperature and relative humidity from the DHT11 sensor
- Read an analog value from the LDR
- Read and interpret values from the microphone
- Write data to a micro SD card

As always, you will need to open the Arduino IDE and either open a new sketch or download the sketches from the links provided.

The first sketch will handle the DHT11 sensor. The DHT11 is a complex device to interface with. Writing code from scratch would be very complex. Fortunately, lots of lovely people have written libraries to help people write code for Arduinos and compatible microcontroller devices like ours. Thanks goes out to them! They make hobbyists lives a lot easier.

Libraries can be downloaded using "the Manage Libraries..." option under Tools in the taskbar. You will need to add two libraries to help us use the DHT11. Search for and install the following:

- Adafruit Unified Sensor
- DHT sensor library

You can download the full sketch HERE.

Reading the comments for each line will give you a good understanding of what is happening. The DHT sensor code refers to separate files downloaded with the libraries. You don't need to understand these to write your own code, just copy the code and adapt it as needed.

The downside to the DHT code is that it interrupts other functions, so be aware of this when trying to update a display every few milliseconds, as an example.

The data received from the DHT sensor is stored in variables (temp and humid). You can call these anything you like, but make sure they are changed in the code and in the variable declaration at the start of the code. The two variables are the sent to the Serial Monitor with some text to make it easier to understand. To access the Serial Monitor, click the magnifying glass icon in the top right corner of the IDE. A reading should appear every 5 seconds stating the text and the data. If a garbled mess shows up instead, make sure the baud rate matches the code.

We set the baud rate to 9600, so find that number in the drop down list on the monitor. Reading the data on a serial monitor isn't very convenient though is it. Attach a Digitiser kit to the project then show the temperature on the display. Click a switch and show the humidity. See Device Sketches in the forums (HERE) for working examples of this and more.

# Coding Basics - LDR

```
1  //Input Pin Variables
2
3  const int LDR = A3;
4
5  //Other Variables
6
7  int light = 0; // variable to hold light level reading
8  int lightLevel = 0; // variable to hold mapped light level
9
10 //Void Setup - runs once at start
11
12 void setup()
13 {
14   Serial.begin(9600); // Initialise Serial Monitor
15   pinMode(LDR,INPUT); // Declaring the LDR pin as an input
16 }
17
18 //Void Loop - repeats forever
19
20 void loop()
21 {
22   light = analogRead(LDR); // save current LDR value to variable
23   Serial.print("Light: "); // print text to Serial Monitor
24   lightLevel = map(light,200,1006,0,100); // maps reading to 0-100
25   Serial.println(lightLevel); // print variable's value and start a new line
26   delay(1000); // delay for 1 seconds
27 }
```

The LDR is very easy to interface with. The LDR returns a voltage between 0 and 5V. The analog pins of the microcontroller can read this as a number between 0 and 1023. We use the analogRead() function for this. (line 22)

We will test the output via the Serial Monitor again. Remember that the analogRead() value can be used for other things as well. We could use it to turn an LED on at low light levels, or to decrease the brightness of the Digitiser display when light levels decrease.

In a real world application, the readings from the LDR won't reach 0 or 1023. To determine the maximum range, we can use the serial monitor. if we use Serial.println(light) instead of lightLevel, we will see the raw value from analogRead(LDR). We can then shine a light directly on it to get the maximum value, and turn the lights off and cover the LDR to get the minimum value.

To make this value more meaningful, we can map it to something more relatable, like 0-100 (a percentage). We can use the map function for this. This ask for the name of the variable we want to map, the expected range of that value, and the range we want to map to. (Line 24).

# Coding Basics - Microphone

The Microphone also requires an analog pin. However, the output of the microphone differs from that of the LDR in that it is in the form of a wave. Sound waves have peeks (higher voltage) and troughs (lower voltage). Our sound signal is centered around 2.5V, so zero noise would read ~2.5V or ~512 through our analog input. The difference between these peaks and the troughs is the amplitude, or volume. So, to find the volume, we need to take many samples in a short amount of time, then take the lowest value from the highest value.

To find these values using code, we can use a for loop to perform a number of readings in quick succession. Within this for loop we first read the raw value from the analog pin connected to the mic. We can then use the min() and max() functions to find the lowest trough value and highest peak value respectively. min() returns the lowest of the two numbers within its brackets. In line 27 we change micMin (which starts at 1024) to micVal (the current mic reading) if it is less than micMin. If this is happening multiple times in a row, this number will end up as the lowest value recorded in the samples. We use max() in the same way with micMax (starts at 0) to find the maximum value recorded from the mic.

Notice we declare the micMin and micMax variables within the loop. This is because we want them to reset every time the loop runs. We then take the minimum value from the maximum value to get the average volume over the sample period.

If you want to increase the sample period, just increase the number of times the for loop runs (currently set to 50). Take note that the rest of the code will wait while these samples are taken, so if you are refreshing display every void loop then you may see a flicker.

```
1  //Input Pin Variables
2
3  const int MIC = A0;
4
5  //Other Variables
6
7  int micReading = 0; // variable to hold mics raw input value
8  int micVolume = 0; // variable to hold the converted mic volume
9
10 //Void Setup - runs once at start
11
12 void setup()
13 {
14   Serial.begin(9600); // Initialise Serial Monitor
15 }
16
17 //Void Loop - repeats forever
18
19 void loop()
20 {
21   int micMin = 1024; // Sets micMin to 1024 before the 50 readings decrease the number
22   int micMax = 0; // Sets micMax to 0 before the 50 readings increase the number
23
24   for (int i = 0; i < 50; ++i) // Perform 50 reads
25     {
26       int micVal = analogRead(MIC); // Read value from microphone
27       micMin = min(micMin, micVal); // Decreases micMin if val is less than it
28       micMax = max(micMax, micVal); // Increases micMax if val is more than it
29     }
30
31   micReading = micMax - micMin; // Finds the average volume
32   micVolume = map(micReading,0,775,0,100); // Maps micReading to a percentage
33
34   Serial.print("Ambient Noise: "); // print text to Serial Monitor
35   Serial.print(micVolume); // print variable's value and start a new line
36   Serial.println("%"); // print text to Serial Monitor
37
38   delay(100); // delay for 100 milliseconds
39 }
```

To interface with the SD card, we need to include some libraries, just like with the DHT sensor. By accessing “Manage Libraries...” install the SD library (built-in by Arduino). Thanks again to those who wrote this code for us to use! The library is based off work by William Greiman and some of this code is influenced by the examples found under File - Examples - SD. These are written by David A. Mellis and modified by Tom Igoe.

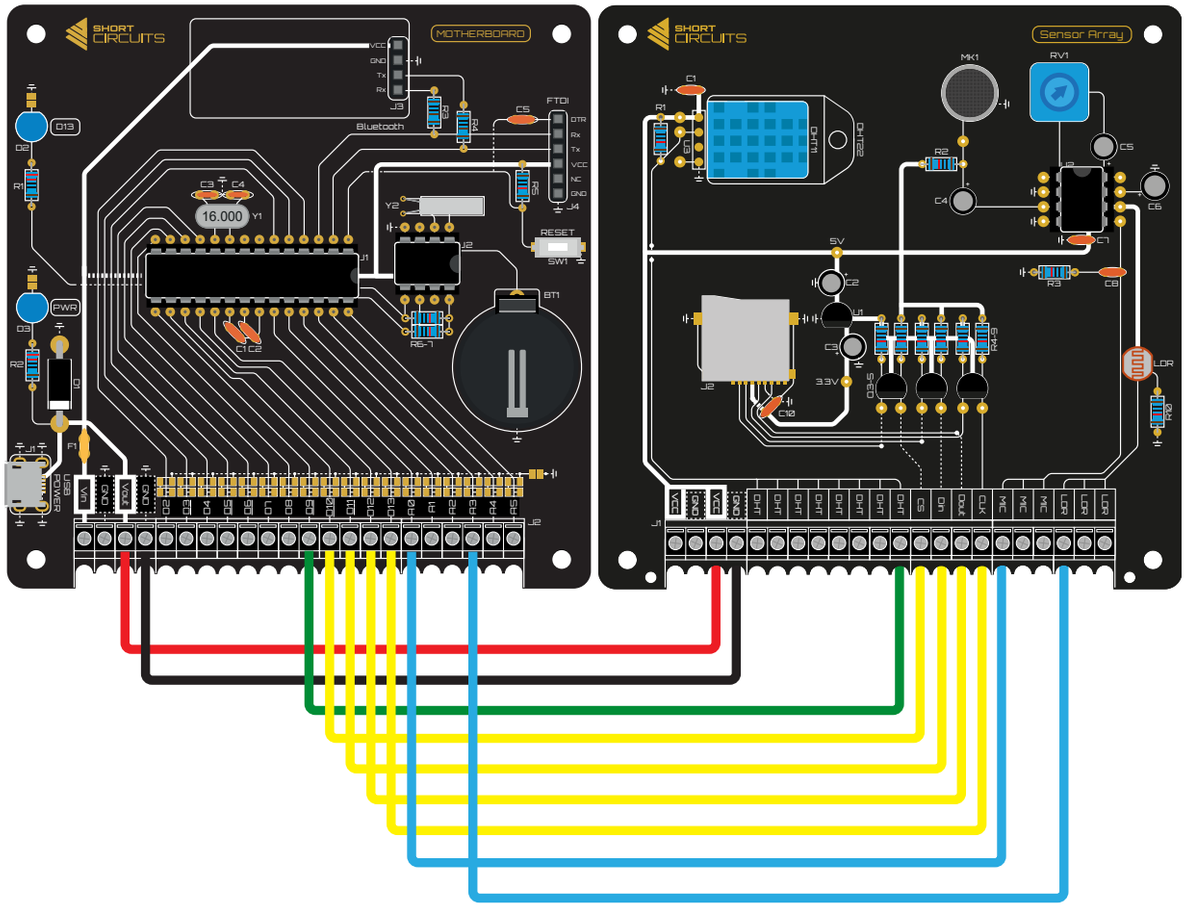
In Void Setup, we must check that there is an SD card slot, with an SD card, connected in the correct way to our microcontroller. This could be done with an if else statement, but the example uses the if(not) while method, so we are going for that one. Notice that the ! before SD.begin indicates a logical “NOT”.

Just like Serial.print writes to the serial monitor, myFile.print writes values to the SD card. The idea of the example sketch is to save 2 pieces of data every 5 mins. The code doesn’t have any useful data to save, but you can use what you have learned to add the data you choose. This could be temperature from the DHT11 and time from the RTC on the Motherboard, for example.

To make it easy to import the data to a spreadsheet, we add a comma between the two pieces of data. This is referred to as comma delimited values and can be easily imported into your favourite spreadsheet software. A new line is made after the second piece of data. So we have two pieces of data on each line separated by a comma. You can have as many pieces of data on a line as you want. Make sure they are all separated by commas. Imagine that a line represents a row in a spreadsheet and the commas separate the columns.

```
1  //Include Libraries
2
3  #include <SPI.h> // Include the SPI communication library
4  #include <SD.h> // Include the SD library
5
6  //Variables
7
8  int data1 = 0; // First variable we want to save to SD
9  int data2 = 0; // Second variable we want to save to SD
10
11 File myFile; // Creates a file object called myFile
12
13 void setup() // Void Setup - runs once at start
14 {
15   Serial.begin(9600); // Initialise Serial Monitor
16   Serial.print("Initialising SD card..."); // Print text to serial monitor
17   if (!SD.begin()) // if statement that checks if an SD card is NOT present
18   {
19     Serial.println("initialisation failed!"); // If not present then print text to serial monitor
20     while (1); // Wait while this is true
21   }
22   Serial.println("initialisation done."); // If there IS a card then print this text and move on
23 }
24
25 void loop() // Void Loop - repeats forever
26 {
27   myFile = SD.open("data.txt", FILE_WRITE); // Open or create file with write permissions
28   if (myFile) // if the file is open...
29   {
30     Serial.println("Data saving..."); // Inform user that the program is saving data to SD
31     myFile.print(data1); // Write first variable to SD
32     myFile.print(","); // Write a comma after the first piece of data
33     myFile.println(data2); // Write the second piece of data
34     myFile.close(); // Close the file to ensure data is saved
35     Serial.println("Data saved"); // Inform the user that the data is saved
36   }
37   else // if the file isn't open...
38   {
39     Serial.println("Error writing to file"); // Inform the user that there was an error
40   }
41   delay(300000); // Delay for 5 mins
42 }
```

# Project Ideas - Data Logger




To build a data logger capable of recording time, date, year, temperature, relative humidity, light levels and sound levels to an SD card, you will need the MOTHERBOARD and the SENSOR ARRAY kits.

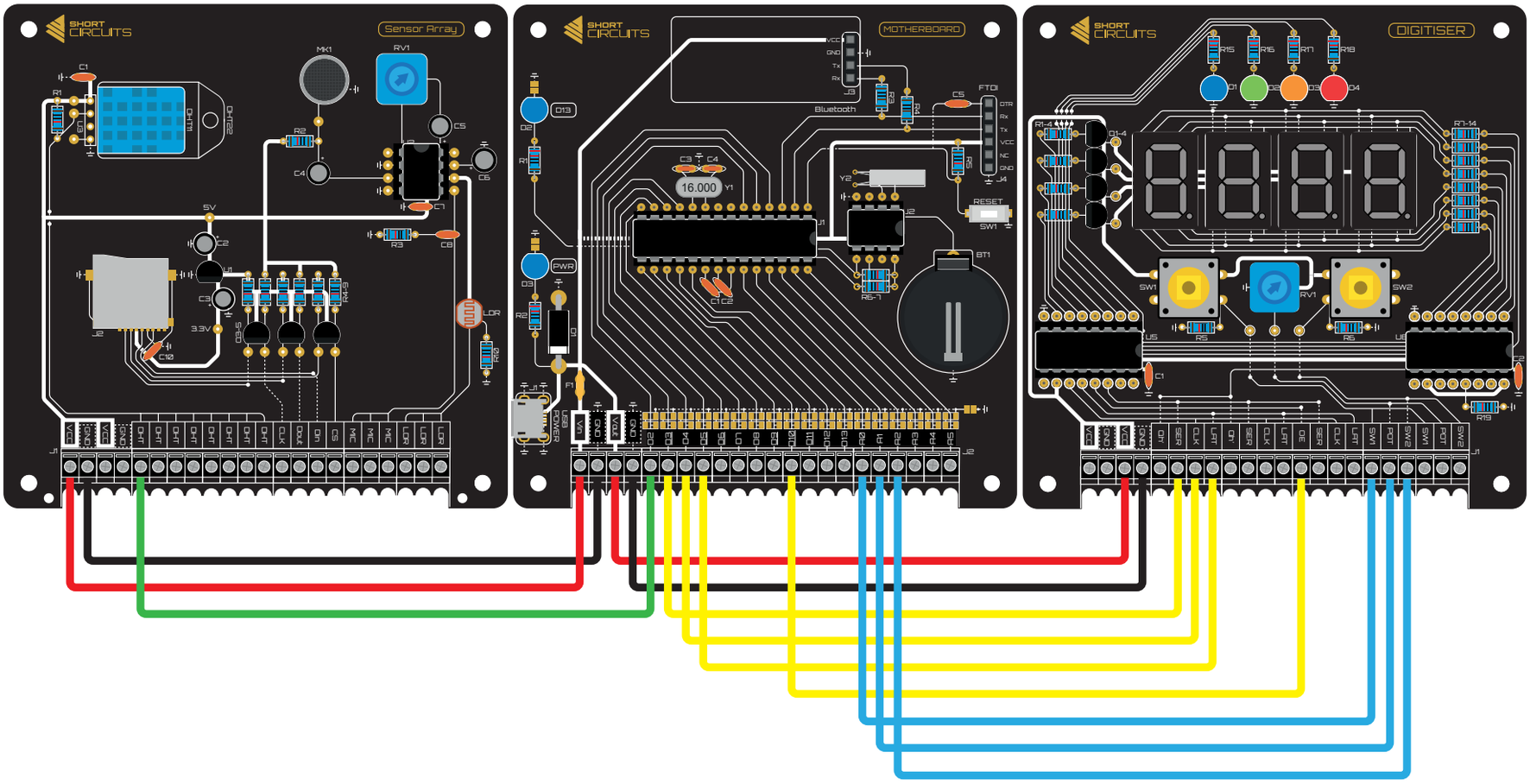
Connect the SENSOR ARRAY up to the MOTHERBOARD as shown in the diagram. If you are going to stack the boards without a case, then the wires need to be as short as possible. A good way to manage this is to insert a wire into one of the screw terminals on the lower board, then stack the boards using the provided standoffs. Cut the wire to length while offering it up to the screw terminal directly above it. Now remove the sheathing from the tip of the wire and push it firmly into the screw terminal. Make sure none of the strands of wire are hanging out the edges. Remove any extra slack if the wire is too long. If you are happy with the length, you can use it as a guide when cutting all the other wires.

If you are going to panel the boards, or stack them in a case, then leave a decent amount of slack. Forward planning is key here. Assemble it without wires, and take some measurements to make sure.

Once connected, head over to the forums and download the DataLogger sketch and upload it using the Arduino IDE and the instructions in the Programming section of the Motherboard's manual.

Remember to put a battery in the holder (BT1) of the Motherboard so the DS1307 can remember the time. That's if you intend on logging time with the other data that is.

# Project Ideas - Environment Display




Adding the DIGITISER to the previous build enables you to add a digital display. This way you can see the current time, temp and humidity. Great for a desktop display.

If you are stacking these, you will get better results if the Sensor Array faces outwards. This will expose the DHT11 sensor to the outside world, and minimise the effect of heat from the other boards on the sensor. If the board is flipped, it may make sense to move the connections around a bit, for neatness and to use less wire. Make sure you change the pin references at the beginning of the code if you do this.

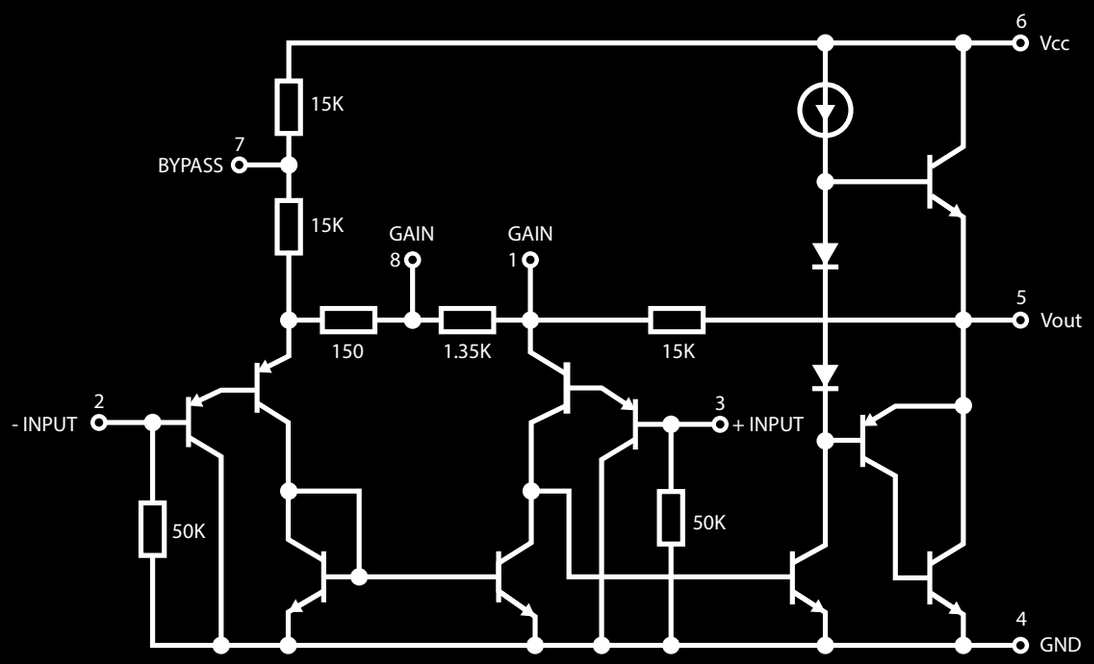
Wire the boards as shown in the diagram above. You can switch outputs on the SENSOR ARRAY and DIGITISER as long as you switch to an IO port that has the same label. Switching outputs on the MOTHERBOARD is fine, just make sure the Potentiometer is connected to analog pin, and Output Enable is connected to a PWM pin. Change the code to match these changes.

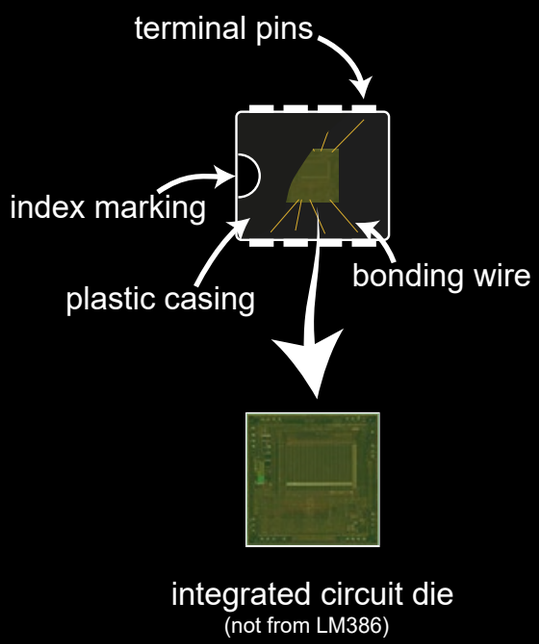
Either write your own code from the hints in each kit's manual, or head over to the forums and grab the EnvironmentDisplay sketch. Upload the sketch using the Arduino IDE and make sure all the functions work. If they don't, check the pin references at the beginning of the code.
Play around with the variables and code to better understand it. Then you can make changes to suit your requirements.

# Component Index

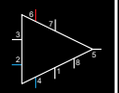
Here you will find information about each component that this kit includes. We have included some of the different sizes and shapes you may find out in the wilderness, different uses for each component, and important data that can be found in datasheets to help you design circuits around them. We have also outlined what happens when these components go pop and how to diagnose and replace them. This information can be used to fix your household appliances, rather than throwing them away. Make do and mend, as they used to say!

# Amplifier - LM386



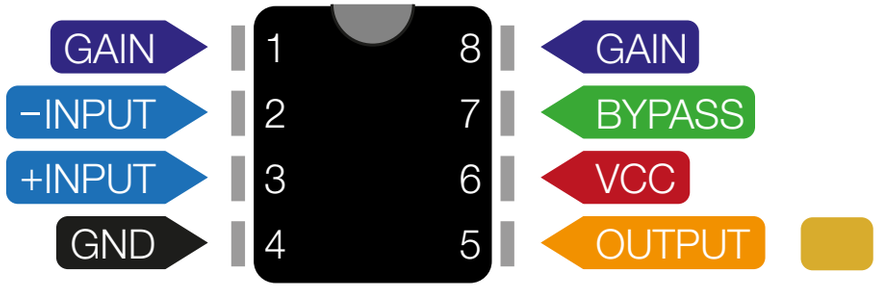

integrated circuit die
(not from LM386)

| | |
|---|---|
| Check Polarity | ✓ |
| Positions | 8 |
| Type | IC |
| Schematic Symbol |  |
| PCB Symbol |  |
| Designator | U |

COMPONENTS

**Important Ratings**

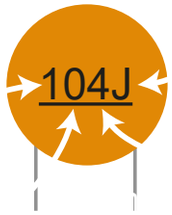
**Pinout**



## Overview

Capacitors are so named for their ability to store a certain capacity of electrical energy (Capacitance), measured in farads (F). A certain amount of capacitance exists between any two conductors that are in close proximity (this can sometimes be seen in an LED matrix in the form of ghosting, and can also cause problems in other sensitive circuits). Capacitors use this in a controlled manner, for various purposes. They usually consist of two conductors separated by a dielectric substance. A dielectric is an insulator (does not conduct electricity) that can be polarised (negative on one side and positive on the other) by an electric field. In this case, the dielectric material is ceramic. A dielectric substance increases the amount of electrical energy that can be stored compared to non-dielectric substances like air.

## Identification

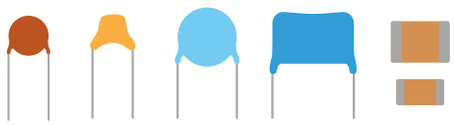



## Quick Reference

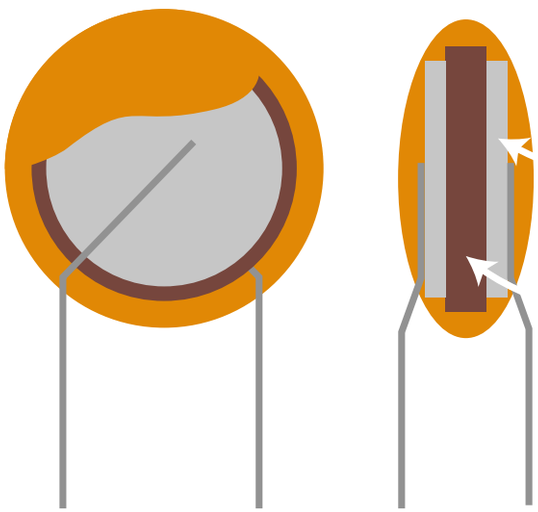
2

passive

## Common Varieties



## Physical Construction



## Important Ratings

## Troubleshooting

Unlike electrolytic capacitors, ceramic capacitors rarely fail in normal use. However, if the voltage exceeds the datasheet's recommended maximum values (breakdown voltage), they can and will produce magic smoke. If they look burnt, or smell burnt, then the charge may have arced across the dielectric layer and will have overheated considerably.

A simple test to see if the capacitor is functioning as it should is to measure it's resistance. As the capacitor charges, the resistance will increase. Before all tests make sure you discharge the capacitor by bridging the leads with a screwdriver. Set your multimeter to Ohms and attach your multimeter probes to each lead. If the resistance climbs to infinity, then the capacitor is functioning as it should. If the resistance reads 0 then the dielectric layer has been compromised and the capacitor will need replacing. This method unfortunately doesn't check its capacitance.

The only reliable way to test a capacitors capacitance is to remove it from the circuit and test it with a multimeter capable of reading capacitance. Set your multimeter to capacitance (Look for the___symbol) and connect the multimeter to the legs of the capacitor. If the value is no longer within the stated tolerance, replace it.

## Common Uses

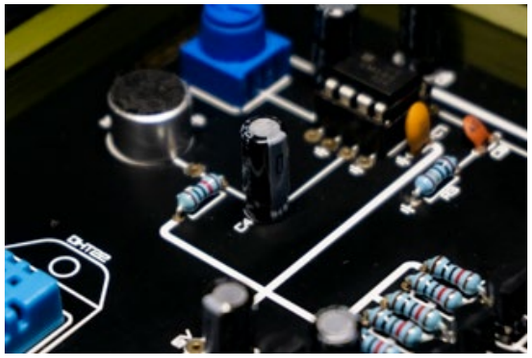
### Decoupling

Capacitors are often used to protect certain components from interference from other parts of the circuit. Most common applications are right next to any integrated circuit (IC). The capacitor acts as a storage reserve. If the voltage drops below the required voltage, the capacitor will use it's stored energy to make up the difference. If the voltage increases above the required voltage, the capacitor absorbs the excess voltage. The microcontroller, or other sensitive device will see a much more even voltage because of this. Decoupling capacitors are placed as close to the sensitive parts of the circuit as possible, for maximum effectiveness. To smooth higher frequencies, you would use a low value capacitor (like a 10nF - 0.1uF ceramic capacitor). To smooth the lower frequency noise, a higher value would be used (often a 1-10uF electrolytic capacitor). Decoupling capacitors feature in most of our kits.

*ATMega328P's Decoupling Capacitors*

### Filtering

Unlike resistors, who's resistance stays constant no mater what frequency of signal that's passing through it, Capacitors are reactive devices that resist higher frequencies less and lower frequencies more. Because of this, they will block DC signals (very low frequencies) and allow AC signals (alternating at higher frequencies) through. This is useful when you need AC and DC in a circuit, but only want AC signals in a certain part. A common example of this is a microphone circuit. The microphone needs a DC signal to power the device, but records AC signals (sound) from the environment. When we pass the AC noise through to an amplifier circuit, we want the AC signal, but not the DC. Adding a capacitor in series will remove the unwanted DC signal. (See the Sensor Array for a working example of this)



*AC Filter on the Sensor Array (electrolytic Capacitor)*

Another form of filter that uses capacitors is an RC filter. The most common RC filters are low-pass and high pass filters. As the name suggests, the low-pass filter lets low frequency signals pass but not high frequencies. High pass filters simply achieve the opposite. This is a passive filter as it uses passive components (resistors and capacitors). The following are the simplest of low-pass and high-pass filters, the equation that governs their properties and a graph to show the typical relationship between frequency (Hz) and amplitude (given in volts).

Time Delay

In an RC circuit, capacitors take time to reach their maximum store of electrical energy when exposed to a voltage source, and to deplete that store when the voltage source is removed. This can be used to create a time delay between turning on a voltage supply and a component receiving the voltage it needs to turn on, or read a logic level high for on a microcontroller for example. We can use a simple equation to work out how much time it will take the capacitor to reach approximately 63% of the supply voltage. This is referred to as the RC time constant and uses the symbol tau (T). In the example circuit the LED will gradually get brighter until the capacitor reaches capacity, following the curve of the graph.

**Low-Pass Filter**


**High-Pass Filter**


**Time Delay Circuit**

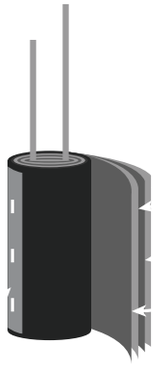
## Overview

Just like ceramic capacitors, electrolytic caps are so named for their ability to store a certain capacity of electrical energy (Capacitance), measured in farads (F). A certain amount of capacitance exists between any two conductors that are in close proximity (this can sometimes be seen in an LED matrix in the form of ghosting, and can also cause problems in other sensitive circuits).

Capacitors use this in a controlled manner, for various purposes. They usually consist of two conductors separated by a dielectric substance. A dielectric is an insulator (does not conduct electricity) that can be polarised (negative on one side and positive on the other) by an electric field. In the case of an electrolytic capacitor, the dielectric material is a non-conductive oxide layer formed on the metal foil of the anode (+). An electrolyte covers the surface of the oxidised layer, acting as an extension of the cathode foil (-).

Electrolytic capacitors are found in much higher values than ceramic or tantalum capacitors. The downside is their tendency to be the first part that fails in a circuit.

## Physical Construction



## Troubleshooting

Electrolytic capacitors are notorious for going bad. Not all are created equally however, and most recommend purchasing Japanese caps. The problem with electrolytic caps is their tendency to dry out. When they dry out the resistance inside goes up and the capacitance falls. As the resistance goes up, so does the heat. This dries the cap out more and eventually the cap will overheat and potentially pop.

To identify a dodgy cap, first look at the top. If it looks somewhat swollen/inflated, then the cap is bad. Not all bad caps have a bulge on the top however. To test a capacitor's capacitance you will need a multimeter with a capacitance setting ( ). This can only be done by removing the cap from the circuit. BE VERY CAREFUL with large caps. They can hold a charge for a very long time, and could hurt or kill you. Short the cap out with a screwdriver or pliers before working on it. After the cap has been removed, touch the multimeter's black lead to the negative leg and the red to the positive leg. The multimeter should show you its capacitance. If it is within the caps tolerance (10% usually) then it should be OK.

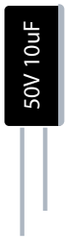
To have a better idea of capacitor health you can use an ESR meter. This will test the capacitors resistance and can be measured while the capacitor is in the circuit. However, these meters are expensive and not often found in a hobbyist's home.

## Quick Reference

2

passive

## Common Varieties




## Important Ratings

## Overview

Resistors do exactly that, they resist the flow of electrons through them. This resistance, measured in Ohms (Ω), is fixed and can be relied on to compliment integrated circuits, divide voltages and protect other components from too much current. But due to the law of conservation of energy, this energy needs to go somewhere. In this case, it is converted into heat. This is why it is important to take note of the power rating of resistors, measured in Watts (W).

Resistors can be made out of many materials, but most commonly metal or carbon film. the film is wrapped around a ceramic core and the whole thing is protected by an insulated layer, usually cream, or blue in colour.

## Troubleshooting

If a resistor is bad you can usually tell. It likely went up in a puff of grey smoke. as soon as you exceed the resistors power rating, it's going to get hot. Fortunately they are cheap and easy to replace. and easy to troubleshoot. Keep in mind that resistance cannot be measured in an operating circuit. Voltage and Current can however. So you could use Ohms law to calculate the resistance.

1. Remove one lead from the circuit
2. Turn your multimeter to its resistance setting
3. Set your multimeter to the lowest range that exceeds the value of the resistor
4. Place the multimeter's probes on each of the resistors leads (polarity doesn't matter) and note the reading.
5. If the value is outside the range of tolerance, then the resistor is bad and needs replacing.

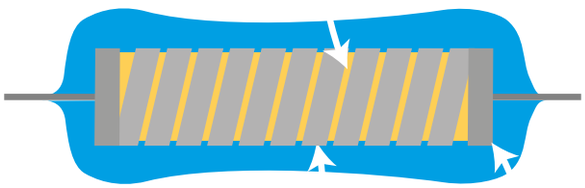
## Quick Reference

2

passive

## Common Varieties


## Physical Construction



## Important Ratings

## Reading Resistor Values

Surface Mount

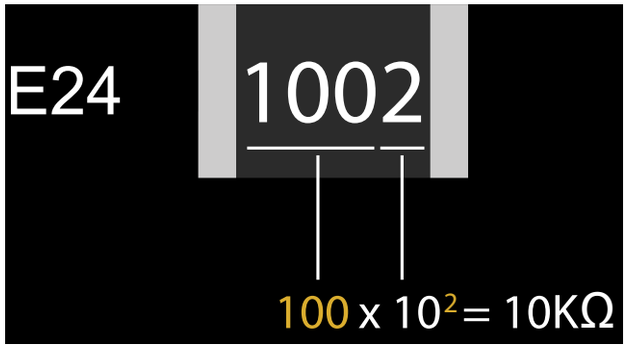
Surface mount resistors use a few coding systems. If you see just numbers, the resistor is probably using the E24 system. If it has a letter at the end then it is probably E96.

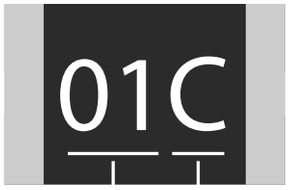
E24

The first two numbers of an E24 resistor represent the 2 most significant digits. The third number represents the magnitude (10 to the power of the third number).

E96

An E96 resistor starts with two numbers that can be looked up in the table, followed by a letter that can be looked up in the other table.



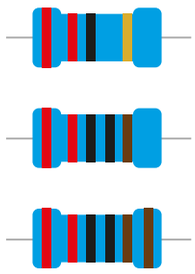

100 100

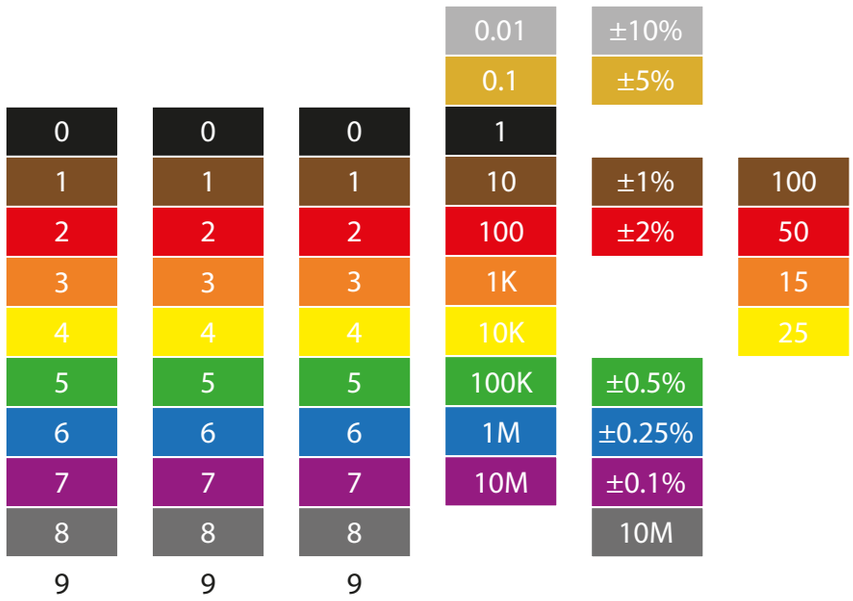
| Value | Value | Value | Value | Value | Value |
|---|---|---|---|---|---|
| 100 | 147 | 215 | 316 | 464 | 681 |
| 102 | 150 | 221 | 324 | 475 | 698 |
| 105 | 154 | 226 | 332 | 487 | 715 |
| 107 | 158 | 232 | 340 | 499 | 732 |
| 110 | 162 | 237 | 348 | 511 | 750 |
| 113 | 165 | 243 | 357 | 523 | 768 |
| 115 | 169 | 249 | 365 | 536 | 787 |
| 118 | 174 | 255 | 374 | 549 | 806 |
| 121 | 178 | 261 | 383 | 562 | 825 |
| 124 | 182 | 267 | 392 | 576 | 845 |
| 127 | 187 | 274 | 402 | 590 | 866 |
| 130 | 191 | 280 | 412 | 604 | 887 |
| 133 | 196 | 287 | 422 | 619 | 909 |
| 137 | 200 | 294 | 432 | 634 | 931 |
| 140 | 205 | 301 | 442 | 649 | 953 |
| 143 | 210 | 309 | 453 | 665 | 976 |

| Value |
|---|
| 0.001 |
| 0.01 |
| 0.1 |
| 1 |
| 10 |
| 100 |
| 1000 |
| 10000 |
| 100000 |

Through Hole

Through hole resistors have 4 to 6 coloured bands which represent digits, a multiplier, tolerance and temperature coefficient. Use the following diagram to work out the resistors value.





**Ohm's Law**

Ohms law is the most basic and useful piece of maths used in electronics. We try to keep things maths free, but this one is unavoidable. It describes the relationship between Resistance (R), Voltage (V), and Current (I).

Ohm's law states that the current through a conductor between two points is directly proportional to the voltage across the two points. So, if the resistance stays the same, then as voltage increases, current decreases, and vise versa. If 2 of the three values (V, I and R) are known, you can easily work out the other using the following triangle.

Ohm's law can be used to calculate voltage drops across components, the current flowing through a circuit, the supply voltage, and the resistance across a component (although some components like an LED do not have a fixed resistance value). This can be useful when diagnosing problems in circuits. If the current is too high, maybe the resistance has dropped across a component for example.


Finding the voltage drop across an LED:
(supply voltage = sum of all voltage drops in a series circuit)


Finding the value of a current limiting resistor:

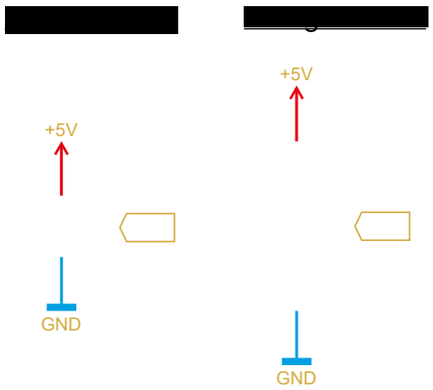

Finding the current draw in a series circuit:
(the current that flows through the resistor is the same as the current that flows through the LED
because they are in series)

## Overview

A potentiometer is a component that offers variable resistance, determined by the position of the wiper on a resistive track. Some potentiometers have twisting knobs, some are long and straight (think of a volume slide or cross fader on audio equipment).

They can be used to give an analog reading between 0V and Vcc. This function can be used to gradually change volume, speed or a multitude of other variables.

A potentiometer acts as an adjustable voltage divider. If you imagine a 10K potentiometer with the knob turned half way, you would in fact have a 5K resistor either side of the center pin. This would give you a value that is half of the supply voltage. In a 5V circuit, this would be 2.5V.




We can calculate Vout by using the voltage divider equation. This asks for three values, the input voltage and 2 resistor values.

$$Vout = Vin \times \frac{R2}{R1 + R2}$$

$$Vout = 5 \times \frac{5000}{5000 + 5000} \qquad Vout = 2.5$$

## Physical Construction

As stated previously, a potentiometer consists of a resistive track with a wiper that moves along this track.


## Troubleshooting

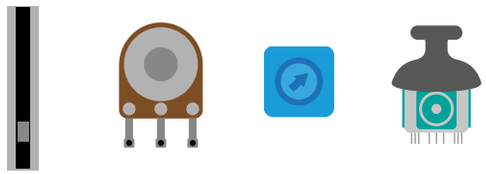
To test whether a potentiometer is working correctly, we can measure the resistance between the pins. First measure the resistance between pin 1 and 3, this should have a value close to the components stated value and within its stated tolerance. In our case, this would be 10K ohms. The potentiometer used in the Digitiser and Sensor Array kits have a tolerance of ±10%, so a value anywhere from 9K to 11K would be in tolerance.

Now we can test the wiper. Turn or slide the pot all the way in any direction. Measure the resistance from pin 1 to 2. This should be nearly 0 ohms or nearly 10K ohms, depending which way you adjusted it. When you turn the pot all the way the other way, this should read nearly the opposite value.

How would a potentiometer fail? Probably the same way any resistor would fail, if too much wattage passed through it, causing it to burn up. If this happened you would expect an open circuit between some of the pins. If there are any issues, swap the potentiometer out for another.

## Quick Reference


## Common Varieties



## Important Ratings

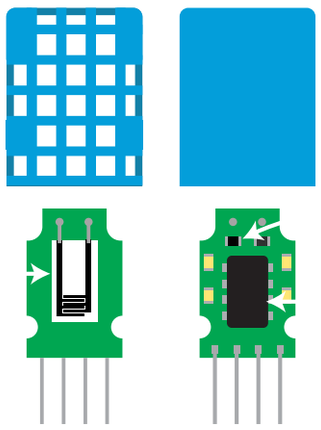
## Overview

The DHT11 is a single data line sensor device that measures temperature and relative humidity. As there is only a single data line, the code has to follow strict timing rules. The microcontroller has to send a signal to the sensor, then wait for a response. This can be tricky to code, but fortunately, kind souls have written libraries for the device. The other downside to this kind of system is that the delay between asking for the data and getting it can interfere with other functions in the code. Fortunately we generally don't need to ask for the data too often. An I2C device would be a better option if this is an issue with your intended use-case.

The DHT11 and others of similar design include a Negative Temperature Coefficient (NTC) Thermistor and a Relative Humidity sensing component. The two sensors are connected to an IC that is used to send the data to the connected microcontroller.

The DHT11 can be swapped out for a DHT22 for better range and accuracy. They use the same footprint and code libraries.

## Physical Construction



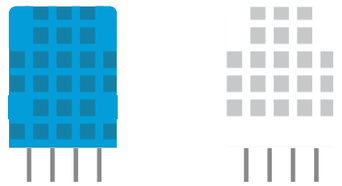
## Troubleshooting

To test whether the DHT is working, we can connect it to a microcontroller and run some code. You could test it in circuit on the Sensor Array, or on a breadboard with the relevant pull-up resistor and bypass capacitor. You could use the "DHT Tester" sketch found in the example sketches, under File in the Arduino IDE. This will give you written feedback in the Serial Monitor telling you if the device is working. You could also use the basic code found in the coding section of the Sensor Array manual. If the Serial Monitor returns 0's instead of the proper temperature and humidity readings, then there is something wrong with the sensor, or your wiring.

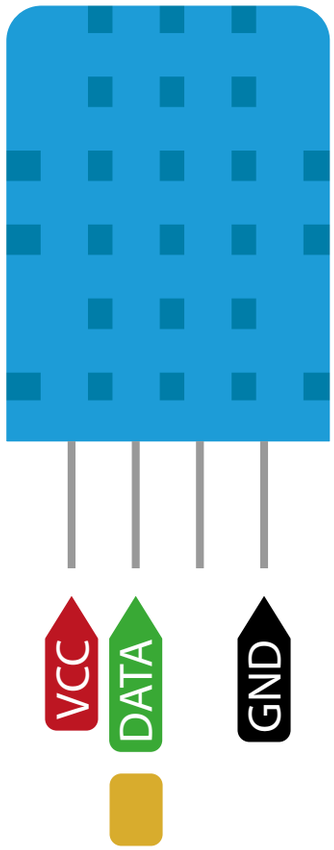
The DHT11's are cheap enough to swap out if they go bad, but the thrifty among you may want to try and replace the NTC temperature sensor or humidity sensor if one goes bad and the other works fine. Probably more hassle than its worth, but a valid learning exercise. The most likely component to fail would probably be the IC, so a replacement DHT11 would be necessary in that case.

## Quick Reference

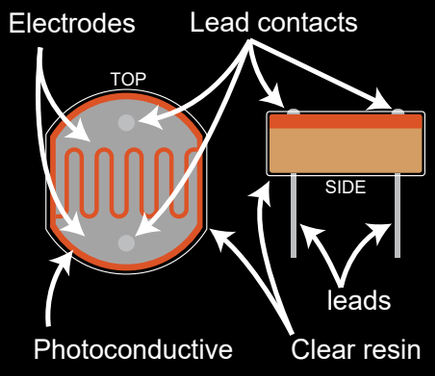

## Common Varieties



## Typical Ratings


VCC
DATA
GND

# Sensor - Light Dependant Resistor




| | |
|---|---|
| Check Polarity | X |
| Positions | 2 |
| Type | passive |
| Schematic Symbol |  |
| PCB Symbol |  |
| Designator | LDR |

| Parameter | Typical Values |
|---|---|
| Max Input Voltage | 90 - 250V |
| Operating Temperature | -60 - 75ºC |
| Light Resistance | 5 - 150KΩ |
| Dark Resistance | 0.1 - 20MΩ |
| Spectral Peak | 500-600nm |

## Overview

## Quick Reference

✓

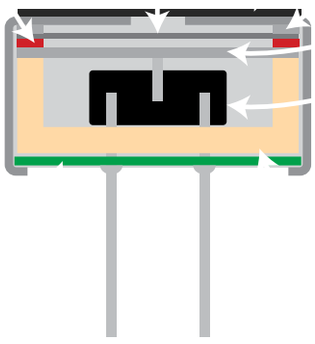
2

active

:

## Common Varieties

## Physical Construction



## Typical Ratings

## Overview

SD cards are a great way to store large amounts of data in a small space. Perfect for integrating into small electronic devices. In cameras and phones etc, SD cards can transfer data very quickly. Unfortunately the communications protocol used to interface with the SD card at these speeds requires a license and lots of legal stuff. So hobbyists are stuck with using SPI. Fortunately SPI is easy to handle for the microcontrollers we use. Although slow in comparison to phones, the kind of data we will save to the SD will be fairly small. So the speed is less important. There are some great Arduino libraries to help us out as well. Check out "SD" in the Arduino IDEs library manager.

## Troubleshooting

There are quite a few places things could go wrong here. Firstly, the SD card needs to be formatted in a way that the software can read and write to it. Only FAT16 and FAT32 are compatible. Formatting it on a Windows PC is the easiest way to do this.

We are also limited to normal SD and SDHC, not SDXC. This means we are limited to a maximum of 32GB. If we know our card is formatted correctly and the size and type are compatible, we can rule that out of the equation.

The most likely thing fault would be in the wiring. Check the pinout diagram on the next page. If you connect the SD card to another microcontroller, make sure the cards CS pin is connected to the microcontrollers SS pin. You can use a different pin for CS but this must be declared in code. Type the pin number in the brackets when using SD.begin(); and make sure to set the microcontrollers SS pin as an output or it won't work.

*Please note the first version of the Sensor Array board has a mistake on the silkscreen. The SD slot's outputs are labeled in reverse. Please check page 5 for details.*

Next we need to check if the card slot is soldered without any bridges. A visual check with a magnifying glass or microscope will do.

We also need to check the 3.3V to 5V logic converter circuitry is working. Connect their outputs to a microcontroller and set them high with code:

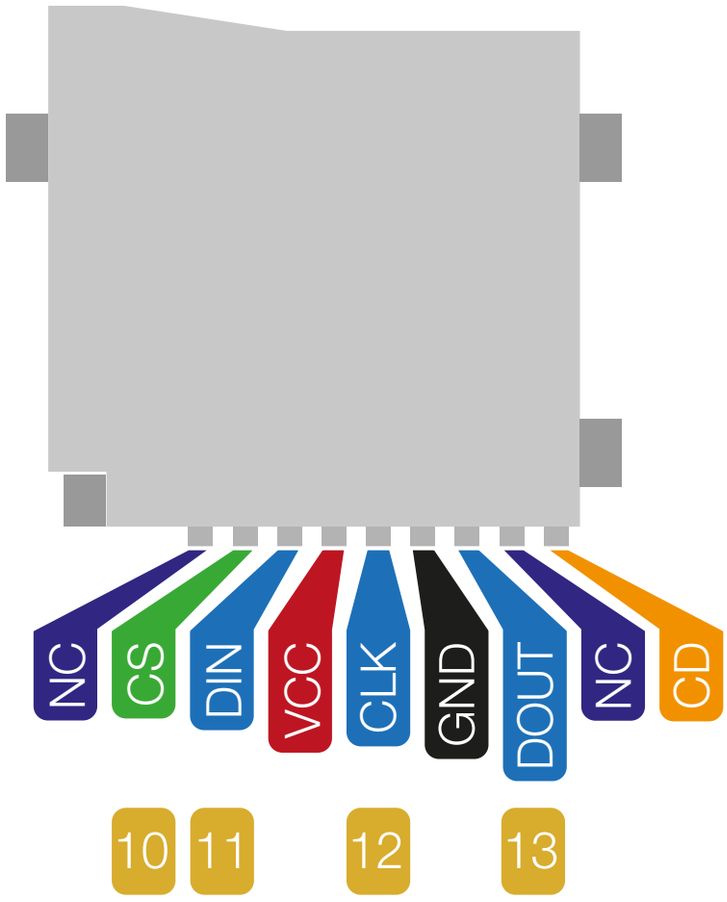
digitalWrite(pin name,HIGH);

We can then check that the data pins of the SD card slot, that use the logic level shifters, are reading 3.3V.

Next we can eliminate the possibility that we have a problem in code by using the SD libraries "ReadWrite" sketch, or any of the other example sketches. These can be found under File - Examples - SD.

## Quick Reference


## Important Ratings


NC
CS
DIN
VCC
CLK
GND
DOUT
NC
CD
10
11
12
13

## Overview

Transistors are everywhere. There are billions in each computer CPU. To learn more about BJT transistors, which were the standard before MOSFETs came along, check out the Digitser or RGB Matrix manuals.

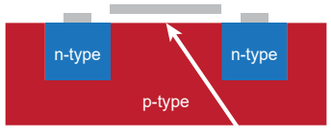
A Metal-Oxide Semiconductor Field Effect Transistor (MOSFET) differs from BJT transistors in that they have an insulating layer between the Gate (equivalent to Base on a BJT) and the Source (Emitter) and Drain (Collector).

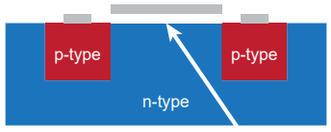
Where the BJT is a current controlled device, where the current at the Collector or Emitter depends on the current applied to the Base, a MOSFET is controlled by the voltage applied to the Gate. A field effect is created by the voltage at the Gate which allows a flow of current between the Source and Drain.

MOSFETs are faster at switching than BJTs so are more suited to high speed data transfer. They also consume less power to operate, so are great for conserving energy when you have billions of them!

MOSFETs come in N-channel and P-Channel flavours, with Enhancement Mode and Depletion Mode versions of each.

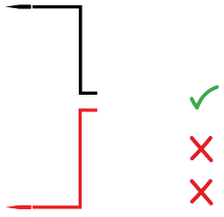
## Physical Construction

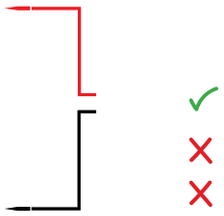



## Troubleshooting

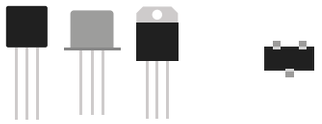
To determine whether a MOSFET has become toast, you can use a multimeter in diode mode ( ). Connect the multimeter to the MOSFET by following the diagram below. The multimeter should read 0.4 to 0.9V. If it reads 0V or Open Loop (OL), then the MOSFET needs replacing.





## Quick Reference


## Common Varieties



## Important Ratings

## Overview

Voltage regulators come in two main types, the linear regulator and the switching regulator. The linear regulator uses a feedback loop to adjust the resistance in the circuit, thus creating a stable output voltage. Switching regulators work by switching the output on and off very quickly to regulate the average voltage output. This works much like a PWM pin on a microcontroller.

The HT7533 is a linear voltage regulator. These are best suited for low power situations as they are less efficient than switching regulators.

## Internal Circuit Diagram

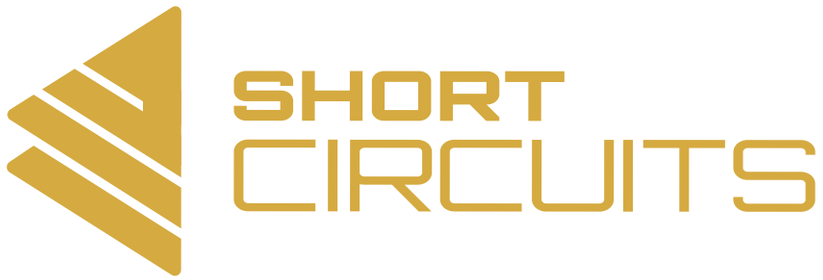
## Troubleshooting

To test a fixed output linear voltage regulator you can simply measure the output voltage. If it is within 0.2V when the output current is well within recommended limits, then the regulator is functioning correctly. If not, swap it out.

## Quick Reference


## Common Varieties

## Important Ratings




This manual was written and designed by Martyn Evans.
The circuit designs are inspired by many different sources with hands on testing and experimentation.
If you recognise anything as your own, and think you deserve a mention,
please feel free to contact admin@shortcircuits.cc and let Martyn know.