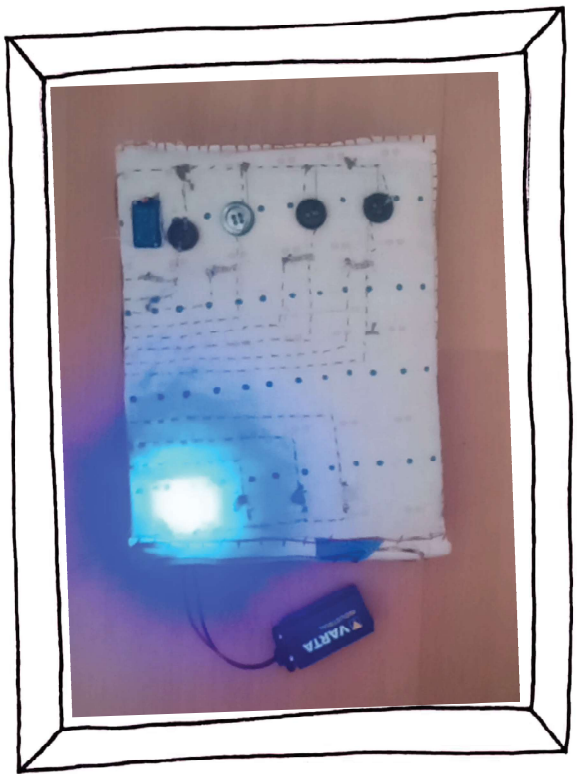
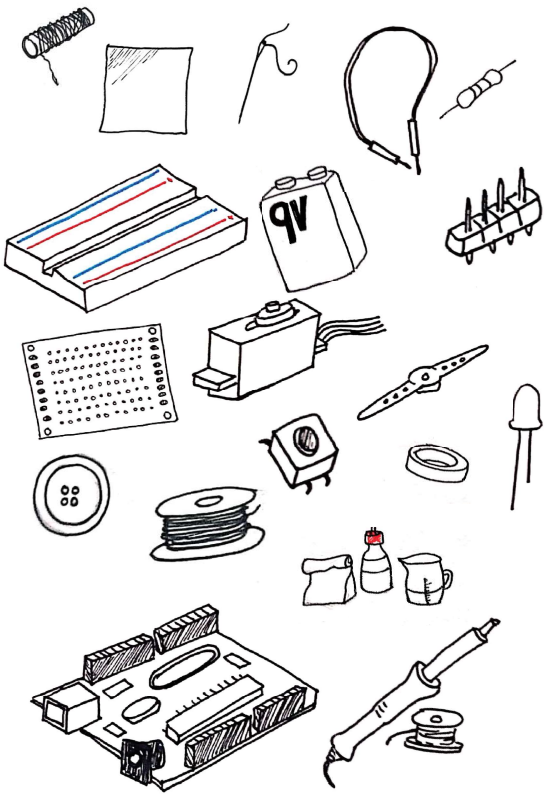
# Secret Diary Box



Create a box which can only be accessed using a secret code! Using an Arduino and integrating the circuit onto fabric you can create a stylish box to safely store all your precious pocessions.

## What you need:




CONDUCTIVE THREAD
NORMAL FABRIC
NEEDLE AND THREAD
SEWING MACHINE

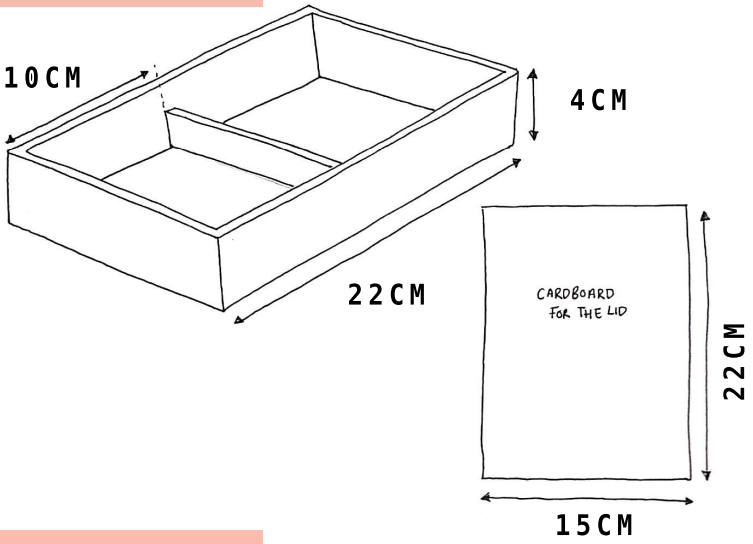
4 LEDS
(DIFFERENT COLOURS)
4 PLASTIC BUTTONS
4 ELECTRONIC BUTTONS
SERVO MOTOR
4 RESISTORS (1K OHM)

ARDUINO UNO
PCB BOARD
THIN INSULATED WIRE
BREADBOARD
JUMPER CABLES
PIN HEADERS
9V BATTERY AND CONNECTOR
SOLDERING IRON AND SOLDER

NEWSPAPER
GLUE, FLOUR AND WATER
PAINT
TAPE
CARDBOARD

# STEP 1

## CREATE YOUR BOX

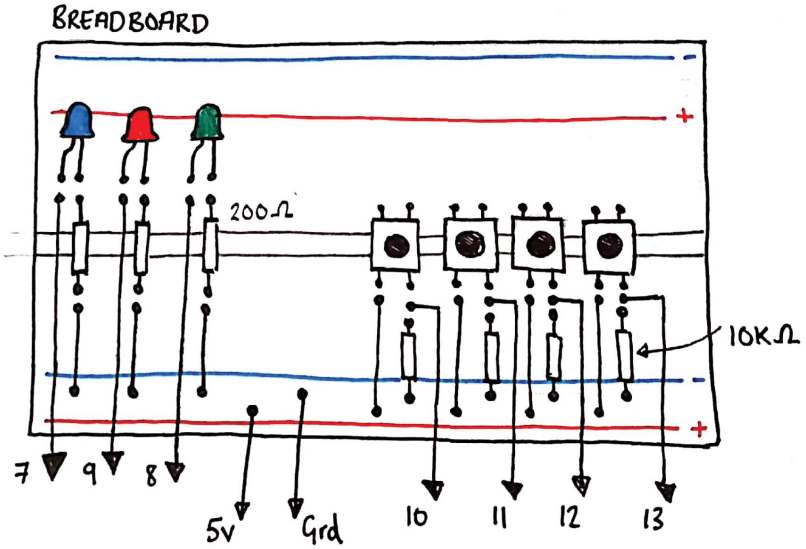



Create your box using cardboard. Cut out the 5 pieces (the base, 4 sides and the middle divider) to create the box on the left, as shown in the illustration. Hold the pieces together using tape. Using glue with a little bit of flour and water, create a paste. Paper mache the box to hold all the elements in place using pieces of ripped up newspaper. Leave to dry then paint. Cut a seperate piece of cardboard for the lid.

# STEP 2

## BUILD A TEST CIRCUIT

Create a prototype circuit using electronic components, a breadboard and jumper cables. Follow the diagram below, connecting the components with the Arduino ports as defined.




## UNDERSTAND YOUR COMPONENTS

Details on how the components work, such as the breadboard, LEDs and buttons, can be found in the 'Understanding your components' section at the beginning of this booklet.

# STEP 3

## UNDERSTAND THE CODE

Take a look at the code that will be used to program the Arduino. This will create the system that will recognise when someone has tried to enter the box. It will cause the green LED to light up if the code entered is correct or flash red if it is wrong.
Open the file in Arduino IDE and take a look at the notes below. Make sure you understand the code before moving on to the next steps. The Coding Dictionary will help you!

# LINK TO CODE FILE

This first section of the code defines all the elements that will be needed later on in the code. This includes any pins of the Arduino connected to a component and defining any initial variables with values.

```
Define the Arduino pin for each of the LEDS (the variable name which is linked to each LED
takes the number of the pin it is connected to)
#define RED 8
#define GREEN 7
#define BLUE 6

Define your secret code as an Array*
const int code[] = {0, 1, 3, 2, 3};
Here it calculates how long your code is
const int codeLength = sizeof(code) / sizeof(int);

sizeof() command gives the size (how
many values there are inside) of an
Array*

The Arduino pin for each button is being define as part of an Array*
const int buttons[] = {13, 12, 11, 10};
Here it calculates how many buttons there are using the length of the Array* set above
const int buttonsLength = sizeof(buttons) / sizeof(int);
This command creates an Array called 'pressed' which has the same number of values in it as
the number of buttons provided by 'buttonLength'. These values can either be true or false.
bool pressed[buttonsLength];

Index defines how many times the code has been run through. This is initially set to 0
int index = 0;
The bool variable* of correct is given the initial value of true.
bool correct = true;
The variable attempt defines the number of attempts that have been tried to crack the
code. This is initially set to 0.
int attempt = 0;
```

The setup section of the code defines the components within the code, this gives the code an idea of what information needs to be received or given to make a component behave in the way we want it to.

```
void setup() {
Here we set the pinMode of the Arduino pins.
These 3 examples are for the LEDs. We want the LED to light up or turn off when it is told
to do so. This is an output* as it gives an action in reaction to command or signal provided
by the code.
  pinMode(RED, OUTPUT);
  pinMode(GREEN, OUTPUT);
  pinMode(BLUE, OUTPUT);

Use the pinMode() command with Arduino.
The format is as follows:
pinMode (Ardunio pin number, INPUT or OUTPUT)

Here a for clause* is used to quickly define the pinMode for all of the buttons. This can be
done because the buttons were defined in an Array* and all buttons provide an electrical
signal when pressed which will be used in the code as an input.
  for (int i = 0; i < buttonsLength; ++i) {
    pinMode(buttons[i], INPUT);
  }
}

Learn more about this notation in the
Code dictionary section.
```

## ARDUINO COMMON ERRORS

- Make sure to have selected the correct device (in this case Ardunino Uno)
- Make sure to have selected the correct port that the Arduino will be connected to
- Make sure to have installed any necessary libaries that are being used in your code (this will be necessary when adding the servo motor.
- Make sure there is a ';' at the end of each command.
- Make sure the names of variables are the same throughout the code. The code is case sensitive!!

This final section of the code, the 'loop' is the main body. This is the part of the code which will continue to be repeated in a loop.

```
void loop() {
Here the Arduino is told to light up the blue LED if the value of 'index' is 0 (no attempts have been made)
  digitalWrite(BLUE, !index);      The digitalWrite() command with Arduino, uses the following format:
                                   pinMode (Ardunio pin number, HIGH or LOW)

 A for clause* is used here to go through each of the buttons to see if any have been pressed
  for (int i = 0; i < buttonsLength; ++i) {

 An if clause* is used here saying that 'if button no.() has been pressed' then continue.
    if (digitalRead(buttons[i])) {
This checks if the same button has just been pressed and if so 'return' meaning ignore this input as it is likely to be an accidental double press.
      if(pressed[i]) return;
If the code does not 'return' it will complete this command, to make the equivalent entry in the array 'pressed' to be true. recording that the button has been pressed.
      pressed[i] = true;

The code assigns bool variable 'correct' to true if the number of the button pressed (identified here as 'i') matches (==) to the corresponding button number in the pin set at the beginning.
      correct = correct && i == pin[index];
This continues to run through until the number of pressed buttons is equal to the length of the code defined at the beginning. This is done using this if clause where index is the number of times it has reached this line of code therefore is equal to the number of buttons pressed successfully.
       if (++index == pinLength) {
Here, if the inputted pin matches the pin defined at the beginning, the code continues to flash the green LED.
         if (correct) {
Here the green LED turns on
           digitalWrite(GREEN, HIGH);
The number of attempts is reset to 0.
           attempt = 0;
The code then pauses here to make sure the green LED is visible to the user.
           delay(1000);
         }
```

else is used here to define the path which the computer will take if the pin entered was not correct.

```
      else {
```

The value of 'attempt' recording the number of attempts tried by the user, will increase by 1

```
      attempt++;
```

The code then checks if the number of attempts is within the limit of 5 attempts the user can make, which in this case is set to 5.

```
      for(int j = 0; j < attempt; j++) {

        for(int k = 0; k < 5; k++) {
```

If so, it will mean that the code was wrong but they can try again, so the device will flash the red LED using the digitalWrite command.

```
          digitalWrite(RED, HIGH);
          delay(100);
          digitalWrite(RED, LOW);
          delay(100);
        }
      }
    }
```

Once either of these two options have been taken and completed, before the code runs back through the loop again, it makes sure that the red and green LEDs are off.

```
              digitalWrite(GREEN, LOW);
              digitalWrite(RED, LOW);
```

It resets the index (the number keeping track of how many times a button was successfully pressed) is reset to 0.

```
              index = 0;
```

correct is reset to its default state of true

```
              correct = true;
            }
```

Finally, for the situation where no button is pressed in the first place, no true value is assigned into the pressed array.

```
          } else {
            pressed[i] = false;
          }
        }
```

A small delay is then included to counteract the incredibly fast speed that the arduino works at, before the loop starts again.

```
        delay(10);
      }
```

## TEST RUN YOUR CODE

Complete a test run of your code,with you Arduino correctly connected to all the components in the bread board. C

# STEP 3

## MODIFY THE CODE

Now that we have understood the code and tested that it runs with the circuit prototype, we can modify the code to work as a secret diary box. We can continue to test it with our temporary circuit. First we will need to include a white LED which flashes each time a button has been pressed. Then we will add a servo motor, which will turn when the correct code has been entered, whcih will eventually act as the lock for our box.

## ADD A WHITE LED

**Hint #1: Steps to include**

- Define the Arduino pin for the LED
- Set the pinMode for the LED
- Include the commands to make the white light flash when a button has been pressed.

**Hint #2: Lines of code to include**

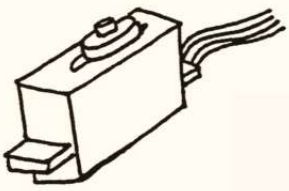
```
#define WHITE 9

pinMode(WHITE, OUTPUT);

digitalWrite(WHITE, HIGH);
delay(100);
digitalWrite(WHITE, LOW);

digitalWrite(WHITE, LOW);
```

**Hint #3: Modified code**

INSERT LINK TO MODIFIED CODE FILE

## ADD A SERVO MOTOR

**Hint #1: Steps to include**



- Install and include the library for a servo motor
- Create an object recognised as 'servo motor'
- Define the Arduino pin for the servo motor (5)
- Define the Open and Closed states of the motor
- Attach the servo to the number corresponding to the Arduino pin
- Setup the servo to be initially closed
- Define the pinMode of the servo
- Make the servo turn when the correct code is given
- Make sure the servo is turned off before resetting for the next attempt

**Hint #2: Lines of code to include**

Below are the different lines of code that are needed to be added to the original code to make the servo motor work. Try to put them into the code in the right place.

```
#include <Servo.h>

Servo servo;

#define SERVO 5

const int Closed = 50;
const int Open = 100;

servo.attach(SERVO);

servo.write(Closed);

pinMode(SERVO, OUTPUT);

digitalWrite(SERVO, HIGH);
servo.write(Open);
delay(4000);
servo.write(Closed);
delay(10);
digitalWrite(SERVO, LOW);

digitalWrite(SERVO, LOW);
```

**Hint #3: Modified code**

INSERT LINK TO MODIFIED CODE FILE

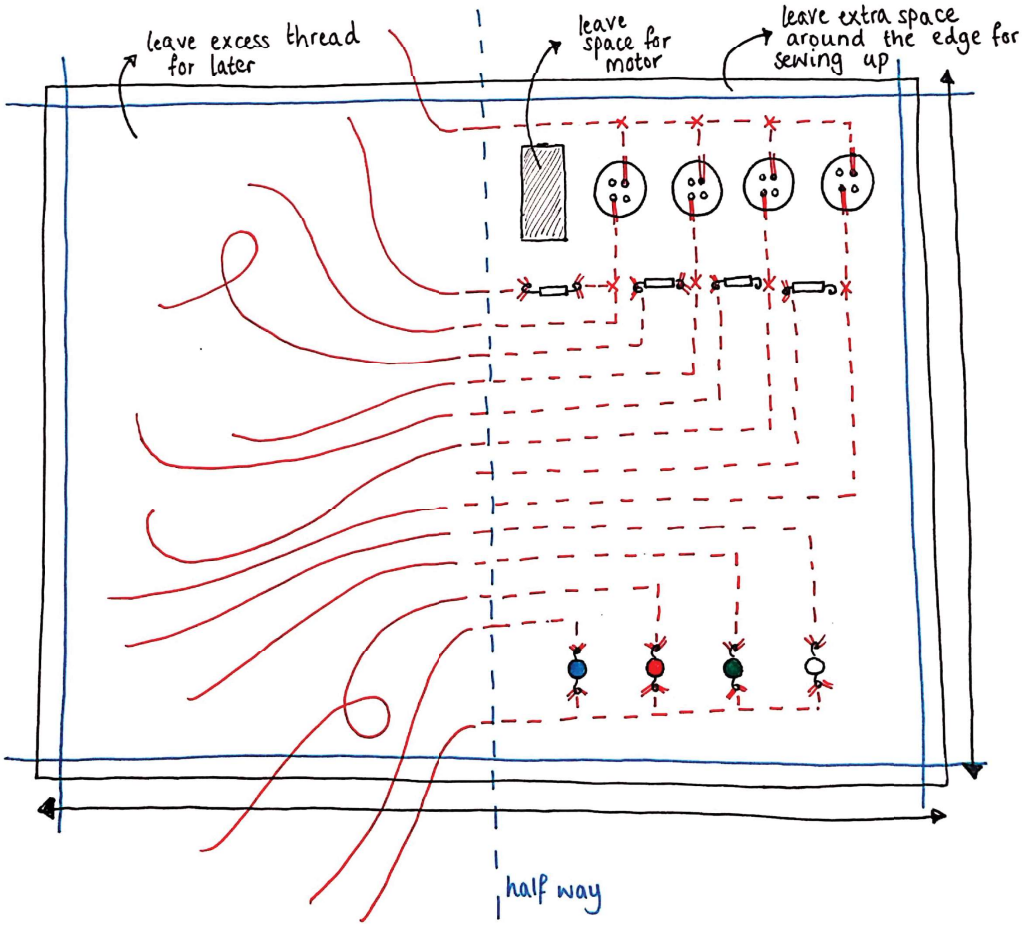
## TEST RUN YOUR CODE

Complete a test run of your code,with you Arduino correctly connected to all the components in the bread board. Compare it to the video.

# STEP 5

## TRANSFER THE CIRCUIT ONTO FABRIC

Steps 5 to 7 are the most complicated steps for this project. Make sure to take your time and regularly test the circuit to make sure there are no errors or bad connections.
To transfer the circuit onto fabric, follow the illustration below. First, sketch out the stitching onto the fabric. You will then be able to identify where to sew on each of the buttons, resistors and LEDS. Do this with normal thread to hold them in place. Then using conductive thread make the connections.
Each connection should be made using a seperate piece of conductive thread.
Make sure to leave a lot of excess thread as shown in the illustration.




## TEST RUN YOUR CODE

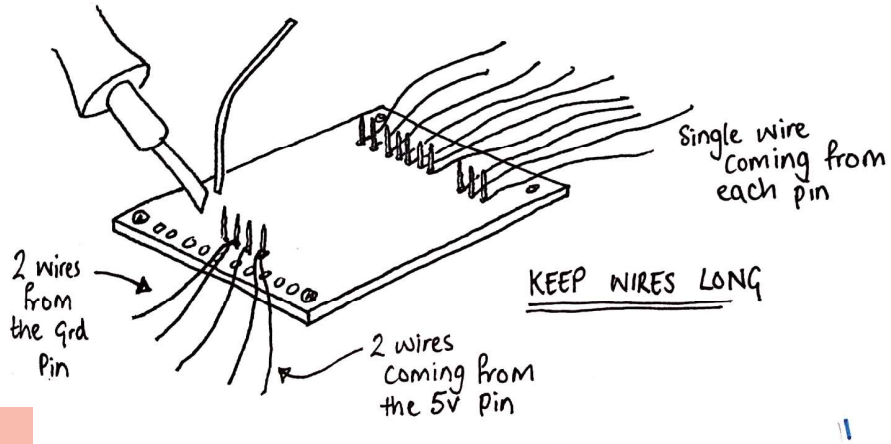
Test your sewn circuit by connecting your Arduino. Use crocodile clips to connect the conductive threads to the correct Arduino ports. Make sure the thread isn't touching any other thread which could cause a short circuit.
If it doesn't work, it could be because:

- Button connections are constantly in a state of 'ON' (by thread always touching)
- Resistors are not connected correctly (they must branch off from the button)
- LEDs are not connected the correct way round.

# STEP 6

## PREPARE THE PCB BOARD

Using solder and a soldering iron, prepare your PCB board with pin headers in the correct locations to match up with the Arduino ports you are using. Connect long pieces of insulated wire to each. Make sure 2 wires come from the Ground and 5V pins.

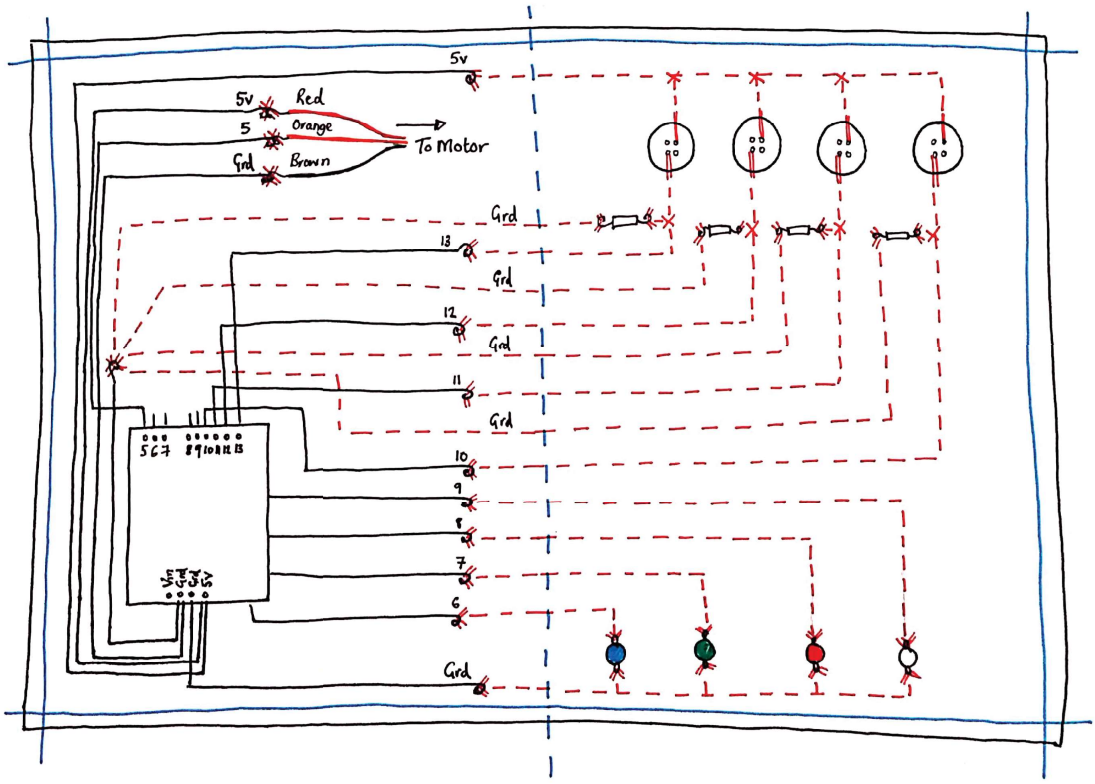



# STEP 7

## CONNECT THE PCB BOARD

Follow the diagram below to connect the PCB board to the sewn circuit. Make sure to:

- Position the PCB so that the Arduino can be connected and will enter the diary box when closed and sit in the smaller section seperated by the divider.
- Sew down the insulated wire. We advise to use a sewing machine with a zigzag stitch. By hand you can use a couching stitch, but this will take a lot longer. It can be a good idea to use take to hold down the wire until it is stitched into position.
- Strip the end of the wire, curl it up and sew it down. Then connect the correct conductive thread to its corresponding pin by forming a connection at this spiraled end of the insulated wire. Make sure no conductive thread overlaps!



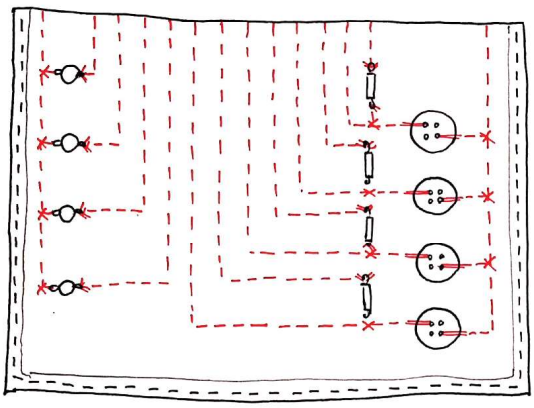
## TEST RUN YOUR CODE

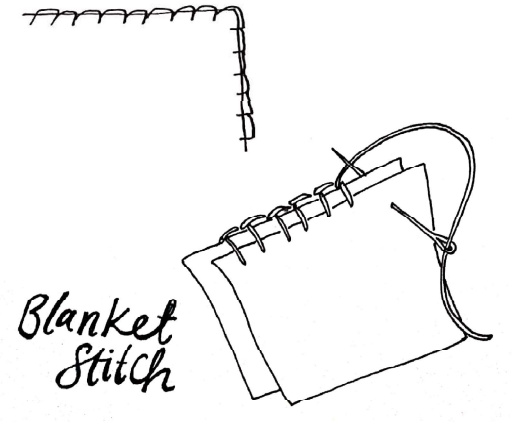
Connect the Arduino to the PCB board and a battery and check the circuit and code continues to work as expected.

# STEP 8

## PREPARE THE LID

Line up the card board to sit between the front and back parts of the fabric. Hand stitch around the edges to enclose the cardboard. You can then fold over the rough edges and tidy them up using a blanket stitch.






## ATTACH THE LID AND FINISH

One the lid, using a stanley knife, cut out a hole for the moto to sit in. Pierce two holes into the lid and two into the side of the box. Connect the lid to the box by threading thread between these holes as shown in the illustration. In the side of the box, cut out an 'L' shaped section. This is for the motor head to lock into when it turns. Check the height needed before cutting. The bottom of the L should be cut at the height of the motor arm when the lid is closed. Finally cut a slit in the bottom side of the box for the battery connection wire to go through.