Paper LED Grid Assignment

Day 0: Transition from previous lesson

- **Discussion Point:**
    - **Purpose of previous circuit:**
        - The previous circuit we created served the purpose of producing a visible pattern on the LEDs, using a control system through the code running on the Arduino.
        - The program on the Arduino controlled LEDs in terms of ROW and COLUMN addressing.

    - **Use of ROW and COLUMN addressing programmatically:**
        - **Advantages:**
            - By using this approach, we are able to extend how many LEDs we can discreetly control on our Arduino, going beyond the immediate input/output pins that are usually available to us.
            - We can see this where we can have different LEDs of a given row turn on or off without affecting the other LEDs.
        - **Disadvantages:**
            - A major disadvantage is that if a LED is on or off for a given column, all the other LEDs of the same column must share the same mode (on or off).
            - This affected certain patterns that could be drawn on the LEDs.
            - The above mentioned problem can be solved by taking advantage of the rate at which we can turn on and off LEDs of a given row to the point that rapid changes will appear to be a consistent image, we will cover this approach in a later lesson.

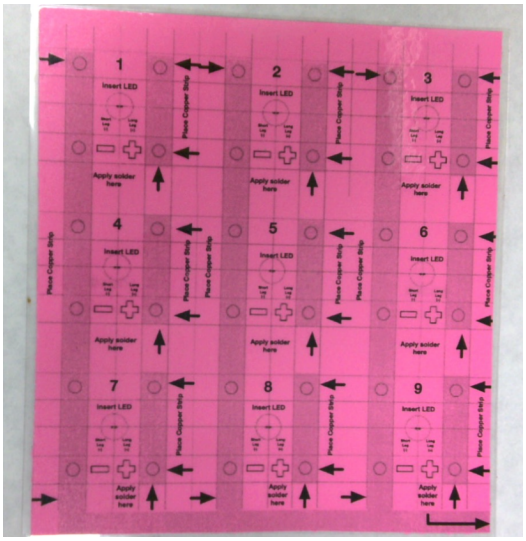
Day 1-2: Assembly Phase - Paper LED Grid and Buttons

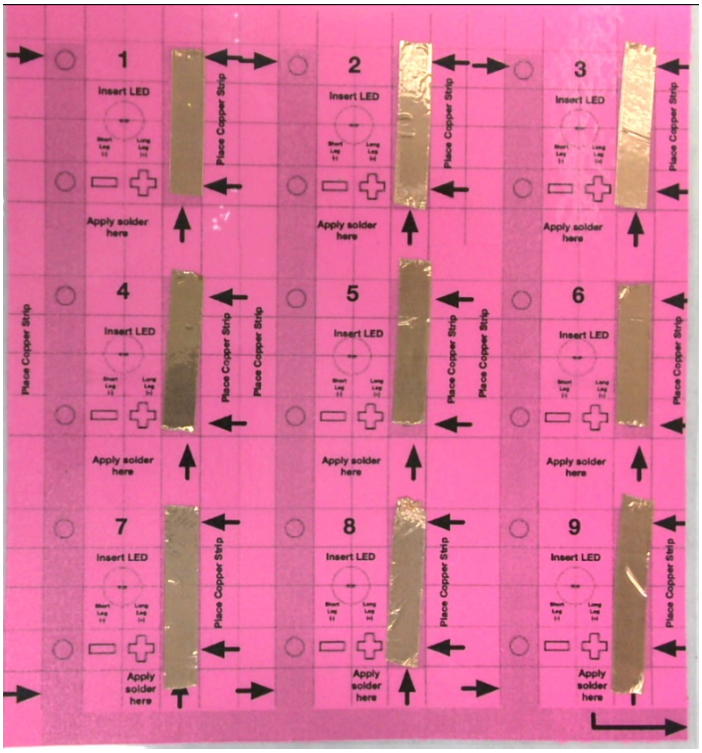
- **Discussion Point:**
    - **Purpose of new circuit:**
        - The following circuit will require that we use more input/output pins of the Arduino, however,by using this approach, we are able to have full control of the exact patterns we can produce through our LED grid, regardless of the relationship an individual LED might have with a given ROW or COLUMN.

    - **Disadvantages:**

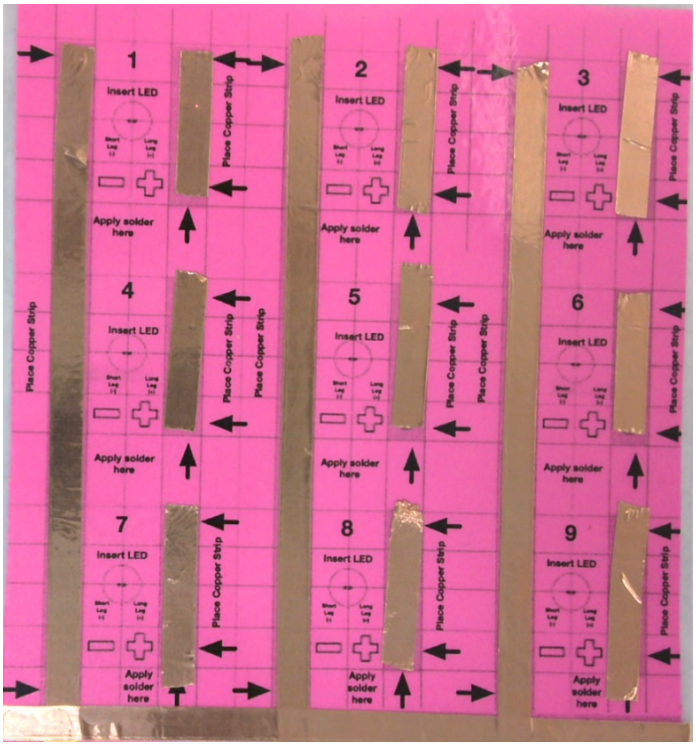
        - Depending on how many LEDs we have connected to each input/output pin of our Arduino, there is an upper limit of how many LEDs we can control.
        - We can effectively control the LEDs by using a common ground (GND) connected to each LED as we will see on our paper LED grid design.
    - **Connection with Numerical Display:**
        - The approach that we are describing here is the same approach that we will use to control our programmable numerical display.
- ## Class Management:
    - Ask students to work in groups of three.
    - When conducting class, proceed through each step of assembly and make sure that everyone has completed the current step before moving on.
- ## LED Grid
    - **Requirement List:**
        -

Part	Quantity
LEDs	9
Paper LED sheet (Printed on cardstock)	1
Multicolored Wire	10
Short Copper Tape Pieces	9
Long Copper Tape Pieces	4
Solder Wire	N/A

    - **Assembly Steps:**
        - Get the paper LED sheet.
        -


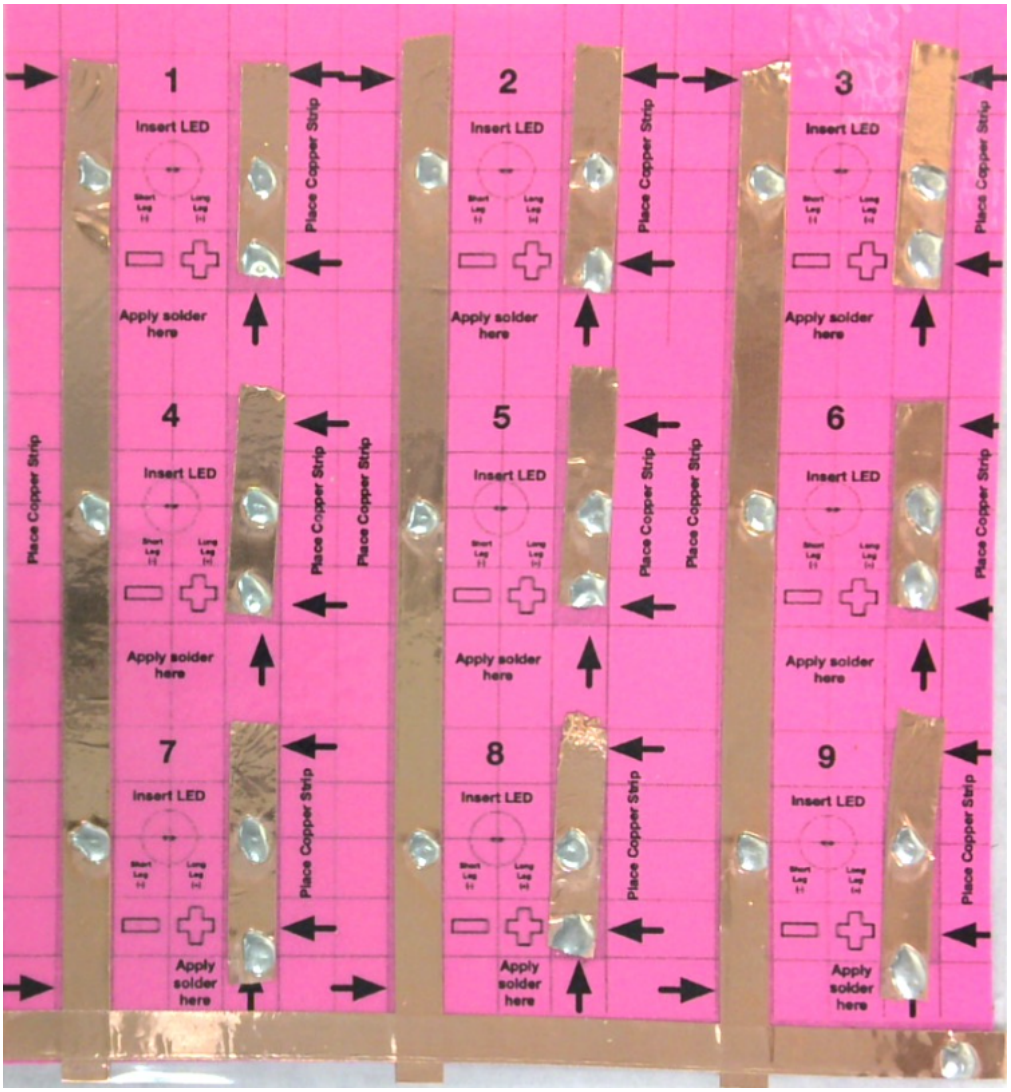
- Get the short copper tape pieces and apply them to each square on the sheet, to the right of each squares “+” symbol (representing the copper tape’s role as a power line.
  - 

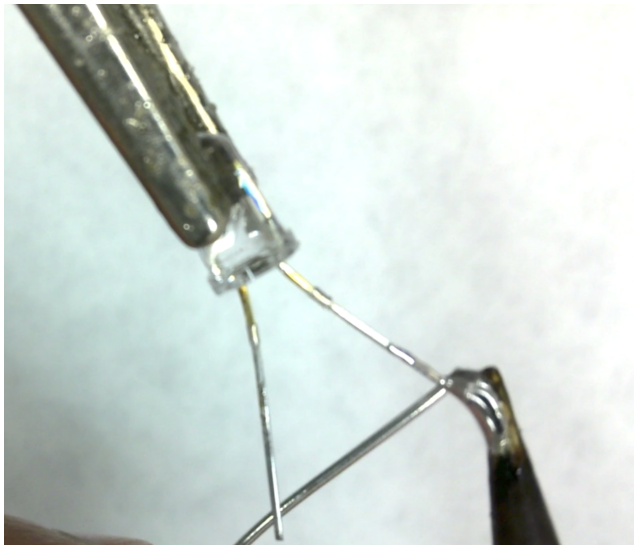
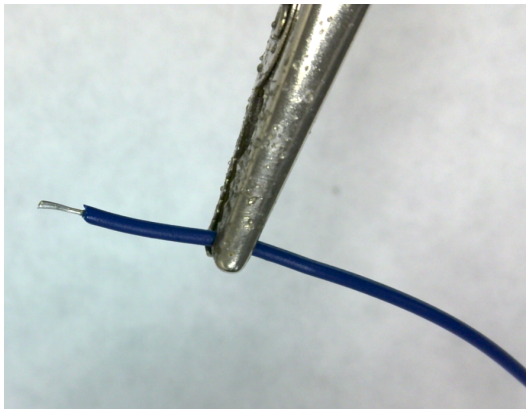
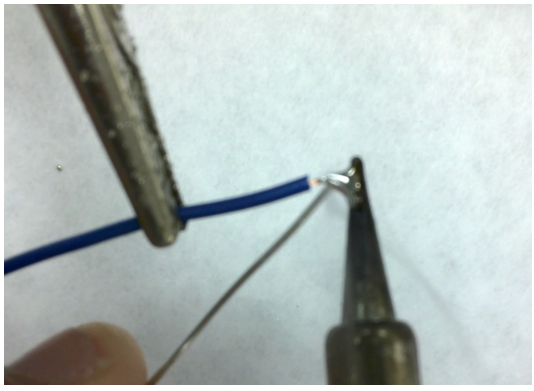
  -
- After applying the short copper tape pieces, we will apply the long copper tape pieces as we have before.

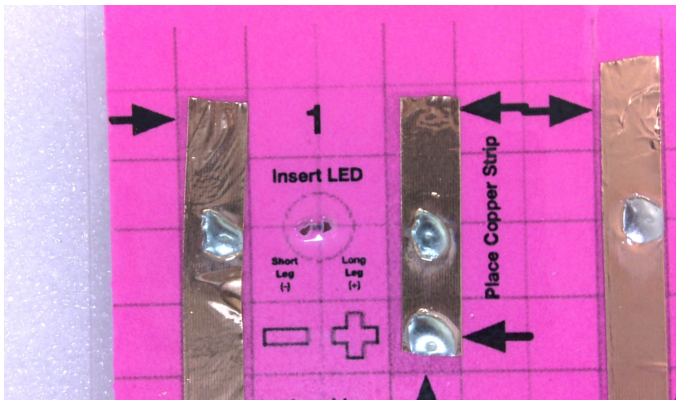
- 

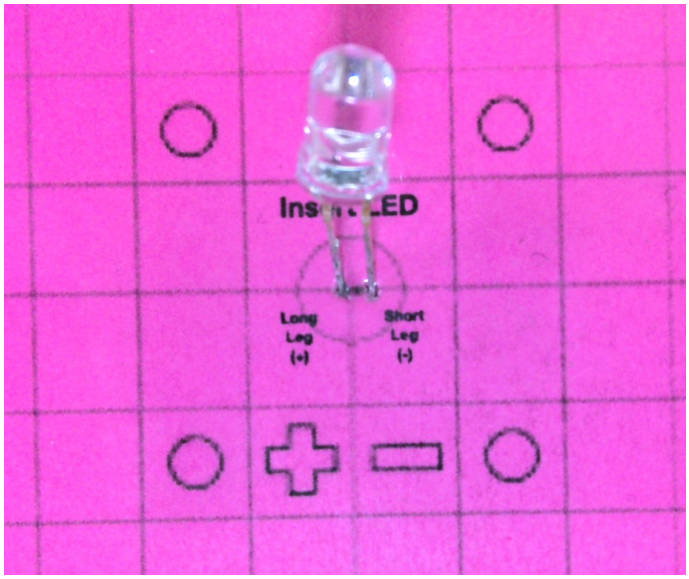
- Next, we will apply solder dots on specific sections of the copper tape in preparation of the following joints between components
  - Jumper wire to copper tape connection.
  - LED leg to copper tape connection.

- **Apply the solder dots as seen below:**

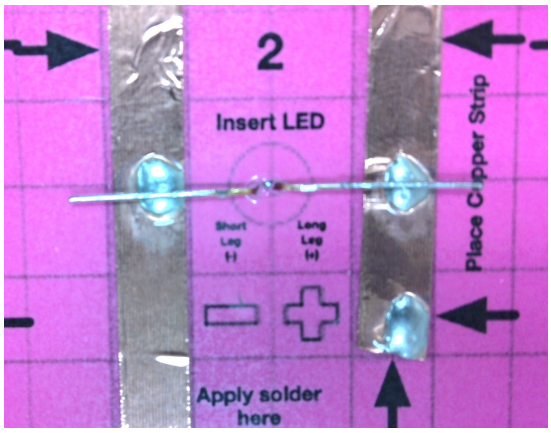

1
Insert LED
Short Leg
Long Leg
Place Copper Strip
Apply solder here
2
Insert LED
Short Leg
Long Leg
Place Copper Strip
Apply solder here
3
Insert LED
Short Leg
Long Leg
Place Copper Strip
Apply solder here
Place Copper Strip
4
Insert LED
Short Leg
Long Leg
Place Copper Strip
Place Copper Strip
Apply solder here
5
Insert LED
Short Leg
Long Leg
Place Copper Strip
Place Copper Strip
Apply solder here
6
Insert LED
Short Leg
Long Leg
Place Copper Strip
Apply solder here
7
Insert LED
Short Leg
Long Leg
Place Copper Strip
Apply solder here
8
Insert LED
Short Leg
Long Leg
Place Copper Strip
Apply solder here
9
Insert LED
Short Leg
Long Leg
Place Copper Strip
Apply solder here

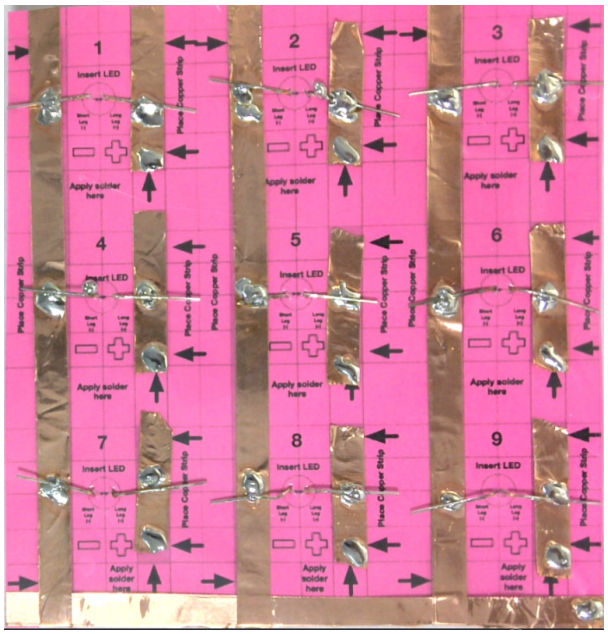
- Next, in preparation of the joints we mentioned above, we will apply solder to the exposed copper ends of the wire and the legs of the LEDs
  - **For the LED leg:**
    - 
    - Link to video for demonstration
  - **For the jumper wire:**
    - 
    - 
    - Link to video for demonstration
-

- Finally, we will solder the connection between the pre-set solder dots on the copper tape and the components we've tinned.
    - **In attaching the LEDs, we will need to puncture a small hole on the front sheet of the LED where our copper tape was applied.**
        - Place a piece of cardboard or cloth behind the sheet and puncture the two holes that are present under "Insert LED".
        - 

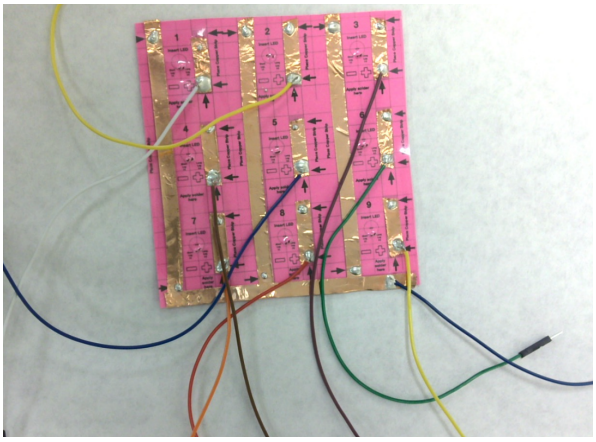
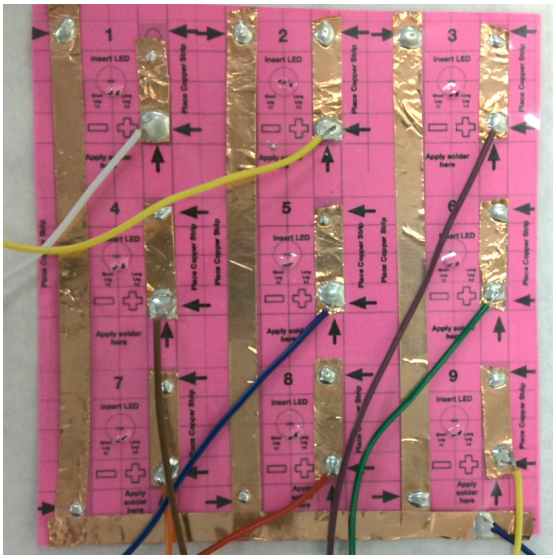
    - **Flip the sheet of paper and place the LED legs where indicated.**
        - 

        -

- Flip the sheet of paper again and fold the legs with the copper tape to its side.
  - 

- Apply solder to your soldering iron and touch the point between the tinned leg and the soldered dot and allow the solder to flow between the components and cool as you remove the soldering iron tip away from it.
  - 

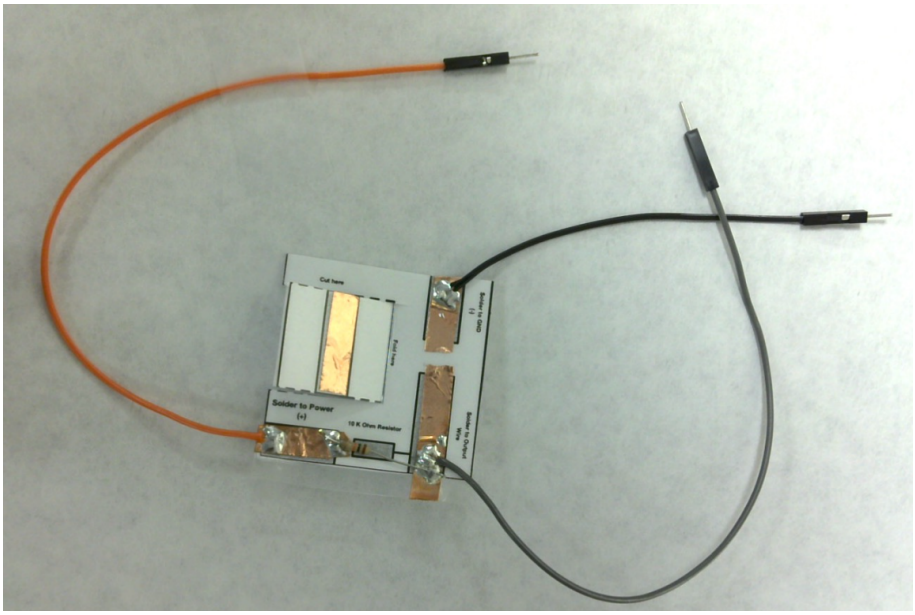
  -

- Similar to our approach to the LED leg-to-copper-tape connection, we will solder our pre-tinned wires onto each of the solder dots on the copper tape.
  - 
  - 

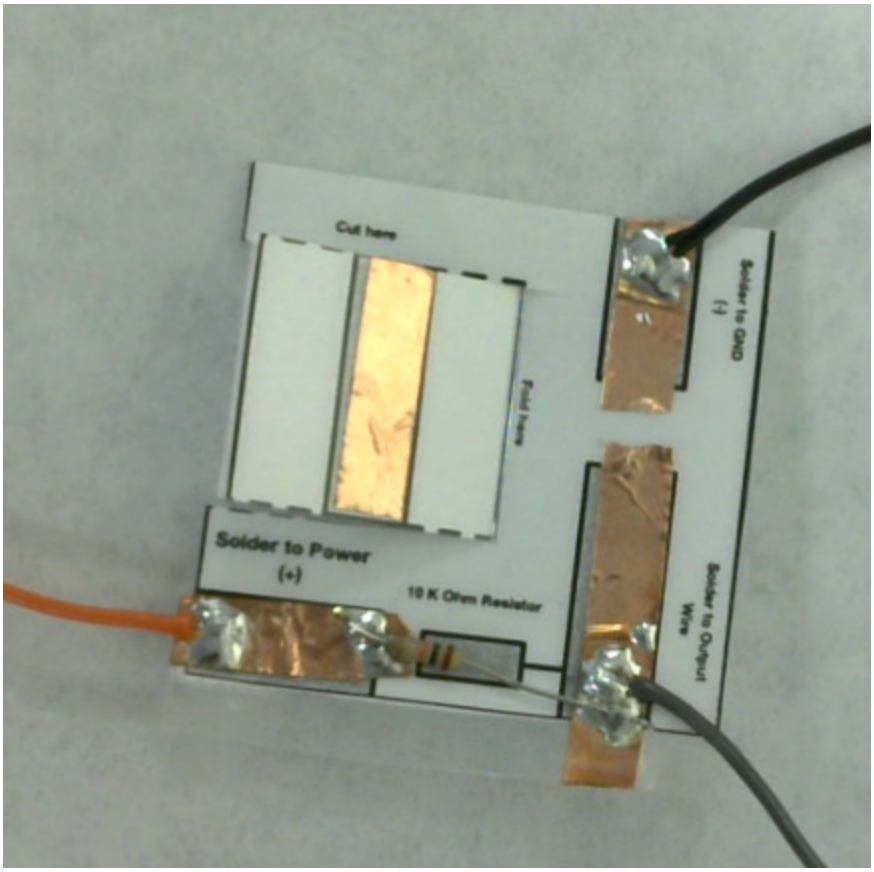
- Pull-up **resistor** buttons
  - Requirement List:
    -

Part	**Quantity**
Paper Pull-up Resistor Sheet (Printed on cardstock)	2
Short Copper Tape Pieces	8
10 K Ohm Resistor	2
Black Jumper Wire	9
Grey Jumper Wire	2

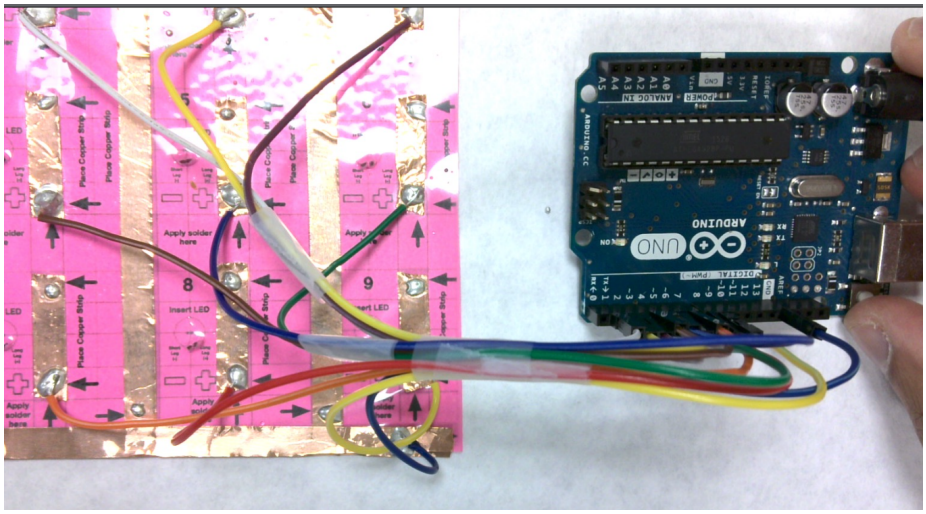
Red Jumper Wire	2
Solder Wire	N/A

- **Assembly Steps:**
  - **End-product:**
    - 

  - Take the paper pull-up resistor sheet and cut it out of the paper sheet. Cut where “cut here” so we can have a flap on the piece of paper.
  - Apply copper tape where indicated.

- 

    - Add solder dots where shown
    - Pre-tin the copper-exposed jumper wire and 10K ohm resistors.
    - Solder the connection between the jumper wire and the solder dots as shown
    - Solder the connection between the resistor and copper tape. Direction does not matter with the resistor.
- ## Connection with the Arduino:
    - **Paper LED Grid:**
        - Connect the black wire connected to the long piece of copper tape and connect it to “GND” on the right side of the Arduino.
        - Connect each colored jumper wire of a specific LED (as indicated by its number) and connect it to the same numbered pinout on the Arduino.

- 
    - **Paper Pull-up Resistor Buttons:**
        - Take a female-to-male jumper wire and connect it to the black wire. Connect the male-end of the jumper wire to one of the three “GND” pinouts on the arduino. Do this for both pull-up resistor buttons.
        - Take a female-to-male jumper wire and connect it to the grey wire. Connect the male-end of the jumper wire to pinout 10 and 11 respectively.
        - Take a female-to-male jumper wire and connect it to the red wire. Connect the male-end of the jumper wire to pinouts “3.3 V” and “5 V” respectively.

Day 3-5: Programming Phase

- ## Discussion Point:
    - We will now move to the programming phase of our paper LED grid assignment.
    - We will overview a specific iteration of our Arduino code, highlighting a key skill to learn/review.
    - **We will cover the following topics:**
        - Programming LED Patterns (Part 1)
        - Assigning LED Patterns to Functions (Part 2)
        - Creating Conditionals for Functions (Part 3)
        - AND Conditions (Part 4)
        - Embedding Sub-conditions (Part 5)
    - We will end with each group using their paper LED grid and buttons to create unique, user-defined animations on their grids, responding to specific button presses.

- **Part 1: Programming LED Patterns**
  - **Discussion:**
    - Our first use for our assembled LED grid is to use it to create patterns.
    - Similar to our previous LED grid, we will use the Arduino to control the LED states to get specific patterns as limited by our 3 x 3 LED grid display.
    - Unlike our previous paper LED grid, we are controlling the LEDs individually, where each LED grid shares a common ground source.
    - Let's think about how we will get specific patterns.
  - **Pseudocode:**
    - Let's imagine we have the 9 individual LEDs that we can control. It would probably go something like this:
    - Pseudocode (don't use in the Arduino):
      - **How to draw X**

```
// Pseudocode isn't real code but its a way to abstract the details
// of what we are trying to code before we try to write it in a real
// programming language like C for instance

// Drawing X on the LED grid

// Row 1 (from the top)
LED_1_state = on
LED_2_state = off
LED_3_state = on

// Row 2
LED_4_state = off
LED_5_state = on
LED_6_state = off

// Row 3 (from the bottom)
LED_7_state = on
LED_8_state = off
LED_9_state = on
```

      - **Here is the real code:**

```
void setup() {
  pinMode(1, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
```

```
  pinMode(10, INPUT);
  pinMode(11, INPUT);

}

void loop() {
  // Draw an X pattern
  digitalWrite(1, HIGH);
  digitalWrite(2, LOW);
  digitalWrite(3, HIGH);

  digitalWrite(4, LOW);
  digitalWrite(5, HIGH);
  digitalWrite(6, LOW);

  digitalWrite(7, HIGH);
  digitalWrite(8, LOW);
  digitalWrite(9, HIGH);

}
```

- ○ **Challenge for students:**
  - Try to write pseudo code that would:
    - Erase a pattern (meaning no LED is on)
    - Draw a plus pattern
- ○ **Code Key (for use by Mentor):**
  - Link

- **Part 2: Assigning LED Patterns to Functions**
  - ○ **Discussion:**
    - Our code was fairly lengthy previously. If you recall from before, we used functions as a way to both simplify our code and allow us to reuse code over and over.
  - ○ Pseudocode:
    - Creating functions (The below is pseudocode, this will not run!)

```
// Suppose we create a function for drawing our two patterns before.
// We could use define these functions somewhere else, encapsulating
//the whole of our instructions. We could then call this function as many
//times as we need, without explicitly writing out the exact instructions
//as before.

drawX(){
        // Drawing X on the LED grid
```

```
        // Row 1 (from the top)
        LED_1_state = on
        LED_2_state = off
        LED_3_state = on

        // Row 2
        LED_4_state = off
        LED_5_state = on
        LED_6_state = off

        // Row 3 (from the bottom)
        LED_7_state = on
        LED_8_state = off
        LED_9_state = on
}

// By this point, all we've done is define our function. We aren't doing //
anything unless we actually call our function.

draw X() // By calling this function, we will turn on and off the
//appropriate LEDs to get the exact pattern we want.
```

- Here is the actual code to create a function for drawing and X pattern:

```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(1, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, INPUT);
  pinMode(11, INPUT);

}

void loop() {
  drawX();
}
// This is our function.
void drawX(){
  // Draw an X pattern
```

```
  digitalWrite(1, HIGH);
  digitalWrite(2, LOW);
  digitalWrite(3, HIGH);

  digitalWrite(4, LOW);
  digitalWrite(5, HIGH);
  digitalWrite(6, LOW);

  digitalWrite(7, HIGH);
  digitalWrite(8, LOW);
  digitalWrite(9, HIGH);

}
```

- **Challenge:**
  - Create a function to draw a plus symbol
  - Create a function to erase all patterns
  - Use the functions in a sequence where x is drawn for a period of 500 ms, erased, then plus is drawn for a period of 500 ms, erased. The program should loop over and over again.
- **Code Key (for Mentor):**
  - Link

- **Part 3: Creating Conditionals for Functions**
  - **Discussion:**
    - We need to think about how we will control our LED grid to produce specific patterns under conditions that we control. This will require us to think about how to create those specific conditions.
  - Pseudocode (it's still not real code):
    -

```
function drawX() …. // Lets say we defined this the same way as
// earlier.

function drawPlus() …. // Lets say we defined this the same way as
// earlier.


If button 1 is pressed, then drawX

If button 2 is pressed, then drawPlus
```

  - Real Code:
    -

```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(1, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, INPUT);
  pinMode(11, INPUT);

}

int delayVal = 250;

void loop() {
  /* This is a comment. Anything that comes after “if( …” and before “..)” is part of a statement that will be evaluated as either true or false. If the statement evaluates as true within the parentheses, the code below the if statement will run. Otherwise, our program will skip over the code block.
 */
  if( digitalRead(10) == LOW ){
    drawX();
    delay(delayVal);
    erase();
   }
}

void drawX(){
  // Draw an X pattern
  digitalWrite(1, HIGH);
  digitalWrite(2, LOW);
  digitalWrite(3, HIGH);

  digitalWrite(4, LOW);
  digitalWrite(5, HIGH);
  digitalWrite(6, LOW);

  digitalWrite(7, HIGH);
```

```
  digitalWrite(8, LOW);
  digitalWrite(9, HIGH);

}

void erase(){
  // Erase pattern
  digitalWrite(1, LOW);
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);

  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);

  digitalWrite(7, LOW);
  digitalWrite(8, LOW);
  digitalWrite(9, LOW);

}
```

- **Code Challenge:**
  - Write code that will have the drawPlus function from before, where it will activate only if we were pressing the button we have connected to our other button as assigned on the Arduino. The conditional should operate similar to how we have it set for drawing X:

```
 if( digitalRead(10) == LOW ){
    drawX();
    delay(delayVal);
    erase();
   }
// You basically need to copy
the code from above and
modify it based on the
conditions described for the
challenge.
```

- **Code Link (For Mentors):**
  - Link

- **Part 4: AND Conditions**
  - **Discussion:**
    - Previously, we had our LED grid respond to our two separate buttons. We can use the combination of our two buttons to produce a third LED grid pattern.
  - **Pseudocode:**
    -

```
// To produce the third option through our combination of button presses
// we need to account for the different conditions that buttons will be
//pressed. The pseudo code below uses a logical AND for the //
statements.

// The case if button 1 is pressed and not button 2.
If button 1 is pressed AND not button 2 is pressed, drawX

// The case if button 1 is not pressed and button 2 is pressed.
If button 1 is not pressed AND button 2 is pressed , drawPlus

// The case if button 1 and button 2 is pressed.
If button 1 is pressed AND button 2 is pressed, drawCircle
```

  - **Real Code:**
    -

```
void setup() {
  pinMode(1, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, INPUT);
  pinMode(11, INPUT);
}

int delayVal = 250;

void loop() {
 /*  In our “if statement” below, we are embedding two logical
statements ( one for reading digital pin 10 and another with digital pin
11) . We connect these two logical statements with “&&” (symbol for
```

```
AND) to evaluate them as a whole, treating the whole as a single
logical statement.*/
  if( (digitalRead(10) == LOW) && (digitalRead(11) == HIGH) ){
    drawX();
    delay(delayVal);
    erase();

  }

}

void drawX(){
  // Draw an X pattern
  digitalWrite(1, HIGH);
  digitalWrite(2, LOW);
  digitalWrite(3, HIGH);

  digitalWrite(4, LOW);
  digitalWrite(5, HIGH);
  digitalWrite(6, LOW);

  digitalWrite(7, HIGH);
  digitalWrite(8, LOW);
  digitalWrite(9, HIGH);

}

void erase(){
  // Erase pattern
  digitalWrite(1, LOW);
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);

  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);

  digitalWrite(7, LOW);
  digitalWrite(8, LOW);
  digitalWrite(9, LOW);

}
```

```
void drawPlus(){
  // Draw plus
  digitalWrite(1, LOW);
  digitalWrite(2, HIGH);
  digitalWrite(3, LOW);

  digitalWrite(4, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(6, HIGH);

  digitalWrite(7, LOW);
  digitalWrite(8, HIGH);
  digitalWrite(9, LOW);
}

void drawCircle(){

  digitalWrite(1, HIGH);
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);

  digitalWrite(4, HIGH);
  digitalWrite(5, LOW);
  digitalWrite(6, HIGH);

  digitalWrite(7, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(9, HIGH);
}
```

- **Challenge:**
  - Modify the code above so that you can account for the two other conditions (when the other button is pressed for plus and when both buttons are pressed for circle).
- **Code Key (for Mentors):**
  - Link

- **Part 5: Embedding Sub-condition**
  - **Discussion:**
    - For our final assignment, we learn how to embed sub conditions within our existing conditions involving our buttons. This will allow us to add 3 more LED patterns.
  - **Pseudo code:**

-

  If button 1 is pressed and button2 is not pressed then draw X. After drawing X if button1 is still pressed, draw H.

  If button 1 is not pressed and button 2 is pressed then draw plus. After drawing plus if button2 is still pressed, draw T.

  If button 1 is pressed and button 2 is pressed then draw circle. After drawing a circle, if button1 and button2 are still pressed, draw C.

- **Real Code:**
  -

```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(1, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, INPUT);
  pinMode(11, INPUT);

}

int delayVal = 1000;

void loop() {

  if( (digitalRead(10) == LOW) && (digitalRead(11) == HIGH) ){
    drawX();
    delay(delayVal);
    erase();

    if(digitalRead(10) == LOW){
      drawH();
      delay(delayVal);
      erase();

    }
```

```
  }
}

void drawX(){
  // Draw an X pattern
  digitalWrite(1, HIGH);
  digitalWrite(2, LOW);
  digitalWrite(3, HIGH);

  digitalWrite(4, LOW);
  digitalWrite(5, HIGH);
  digitalWrite(6, LOW);

  digitalWrite(7, HIGH);
  digitalWrite(8, LOW);
  digitalWrite(9, HIGH);

}

void erase(){
  // Erase pattern
  digitalWrite(1, LOW);
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);

  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);

  digitalWrite(7, LOW);
  digitalWrite(8, LOW);
  digitalWrite(9, LOW);

}

void drawPlus(){
  // Draw plus
  digitalWrite(1, LOW);
  digitalWrite(2, HIGH);
  digitalWrite(3, LOW);

  digitalWrite(4, HIGH);
  digitalWrite(5, HIGH);
```

```
  digitalWrite(6, HIGH);

  digitalWrite(7, LOW);
  digitalWrite(8, HIGH);
  digitalWrite(9, LOW);
}

void drawCircle(){

  digitalWrite(1, HIGH);
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);

  digitalWrite(4, HIGH);
  digitalWrite(5, LOW);
  digitalWrite(6, HIGH);

  digitalWrite(7, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(9, HIGH);
}


void drawH(){

  digitalWrite(1, HIGH);
  digitalWrite(2, LOW);
  digitalWrite(3, HIGH);

  digitalWrite(4, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(6, HIGH);

  digitalWrite(7, HIGH);
  digitalWrite(8, LOW);
  digitalWrite(9, HIGH);
}


void drawT(){

  digitalWrite(1, HIGH);
  digitalWrite(2, HIGH);
```

```
  digitalWrite(3, HIGH);

  digitalWrite(4, LOW);
  digitalWrite(5, HIGH);
  digitalWrite(6, LOW);

  digitalWrite(7, LOW);
  digitalWrite(8, HIGH);
  digitalWrite(9, LOW);
}

void drawC(){

  digitalWrite(1, HIGH);
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);

  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
  digitalWrite(6, HIGH);

  digitalWrite(7, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(9, HIGH);
}
```

- **Challenge:**
  - Change the code above to do the following:
    - If the second button is on and the first button is off, call the drawPlus function. If the second button is still pressed after some time, call the drawT function.
    - If both buttons are pressed, call the drawCircle function. If both buttons are pressed after some time, call the drawC function.
- **Code Link (for mentors):**
  - Link

Final Activity:

- **Discussion:**
  - Below is some code that will help us to create patterns in a more direct way.
- **Code:**

```
int M = 3;
int N = 3;


void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(1, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, INPUT);
  pinMode(11, INPUT);

}

//X Pattern
int x_pattern[3][3] = {
   {1, 0, 1} ,   /*  initializers for row indexed by 0 */
   {0, 1, 0} ,   /*  initializers for row indexed by 1 */
   {1, 0, 1}   /*  initializers for row indexed by 2 */
};

//T Pattern
int t_pattern[3][3] = {
   {1, 1, 1} ,   /*  initializers for row indexed by 0 */
   {0, 1, 0} ,   /*  initializers for row indexed by 1 */
   {0, 1, 0}   /*  initializers for row indexed by 2 */
};


void loop() {


  drawPattern(x_pattern); //



}



void drawX(){

   digitalWrite(1,x_pattern[0][0]);
```

```
  digitalWrite(2,x_pattern[0][1]);
  digitalWrite(3,x_pattern[0][2]);

  digitalWrite(4,x_pattern[1][0]);
  digitalWrite(5,x_pattern[1][1]);
  digitalWrite(6,x_pattern[1][2]);

  digitalWrite(7,x_pattern[2][0]);
  digitalWrite(8,x_pattern[2][1]);
  digitalWrite(9,x_pattern[2][2]);

}

void drawPattern(int arr[3][3]){


 int i, j, k;
 /*
 These are the variables we will use
 to keep track of the changes we need
 to consider as we iterate through each
 element of our 2D array.
 */



 k = 1;
 /*
 We will initialize the value of k at 1. We will use k to
 keep track of which pin we are turning on.
 */
 for (i = 0; i < 3; i++){
   /*
   i will serve as a value to keep track of which row
   of the array we're accessing the values from. The value of
   i will increase by 1 each time we go through the loop.
   */
   for (j = 0; j < 3; j++){
     /*
     j will serve as a value to keep track of which column
     of the array we're accessing the values from. The value of
     j will increase by 1 each time we go through the loop.
     */

     digitalWrite(k,arr[i][j]);
     /*
     Here we will call the digitalWrite function and
     in place of "LOW" or "HIGH", we will use the array value
     specified by i and j in our 2D array.
```

```
    */

    k = k + 1;
    /*
    When we've gone through 1 iteration of turning on
    a single LED, we will increase the value of k by 1 so when
    we return to this same point in the next loop, it will turn
    on the next pin we specify (moving from pin1
    to pin 2 to pin 3...)

    */

   }
  }

}
```

- For our final activity, we will ask students to create a custom animation for each button and button combination using the approaches we've described above. When students complete their animations, each student group will exchange their code with one another to review and comment on how their code is organized.