Part II – Digital Clock

The digital clock project uses an Arduino to control a decoder, which in turn outputs a signal to control the 7 Segment Indicator to display numbers. The digital clock project integrates two buttons to reset the loop and pause the counting, respectively.

Hardware Connection:

Four digit output pins (**unitInputs[4]**): Connect to the BCD input pins of a device (e.g., digital tube) that displays single digits, corresponding to pins D9, D10, D11, D12.

Four tens outputs (**tensInputs[4]**): connected to the BCD input pins of a device (e.g., digital tube) that displays tens digits, corresponding to pins D5, D6, D7, D8.

Two switches (switchPin1 and switchPin2): connected to pins D2 and D3 to control the displayed value.

When wiring the 7 digits are anode digits the middle pin has to be connected to vcc the ppt shows a cathode digit wiring delayed some time.

Program Design:

We have written the BCD array of ten digits from 0-9. Use the delay(1000) function to pause the programme for 1 second. For the reality of tens and single digits, we use the following code, which is implemented by dividing the number by the remainder.

The counting of the seconds uses a mathematical remainder idea to separate the digits of the seconds from the tens for display purposes.

unitNum: Computes the first digit of Num (Num % 10).

tensNum: Calculates the tens digit of Num ((Num / 10) % 10).

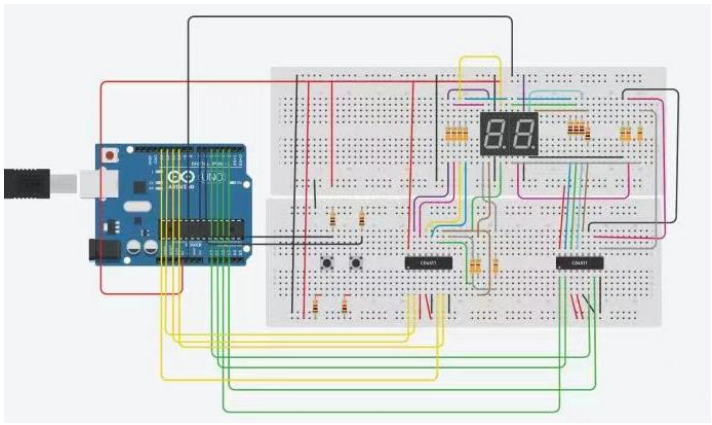
Set the state of the-----D input pin of the display to show the digit and tens digit according to the BCD array.

```
  // Calculate tens and units digits
  int unitNum = Num % 10;
  int tensNum = (Num / 10) % 10;

  // Display tens digit
  for (int c = 0; c < 4; c++) {
    digitalWrite(tensInputs[c], BCD[tensNum][c]);
  }

  // Display units digit
  for (int c = 0; c < 4; c++) {
    digitalWrite(unitInputs[c], BCD[unitNum][c]);
  }

  delay(1000);
}
```



Switch Logic Design:

First, we are A17 group, the provisions of the display value from the beginning of the 16 incremental to 17, and then return to zero to start incremental, and thereafter, if you do not operate the switch, the value has been incremented to 99.

When the switch is pressed, the pin to which the switch is connected outputs a high level, and in the code we use this feature to determine the state of the switch.

```
// Check the state of switch 1
bool switch1State = digitalRead(switchPin1);
// Check the state of switch 2
bool switch2State = digitalRead(switchPin2);

// Handle switch 1
if (switch1State == HIGH) {
  // Switch 1 is pressed
  switch1Pressed = true;
} else {
  // Switch 1 is released
  if (switch1Pressed) {
    // Switch 1 was pressed, now released
    switch1Pressed = false;
    Num = 17; // Reset Num to 0 if switch 1 is pressed
  }
}
```

```
// Handle switch 2
if (switch2State == LOW) {
  // Switch 2 is pressed
  switch2Pressed = true;
} else {
  // Switch 2 is released
  if (switch2Pressed) {
    // Switch 2 was pressed, now released
    switch2Pressed = false;
  } else {
```

Self-locking switch 1, is defined as a reset switch. Set switch1Pressed to true when switchPin1 is pressed (i.e., high), and when switchPin1 is flicked up (i.e., low), the numeric Num is back to 0.

Self-locking switch 2 is defined as a pause switch。In the code, the **switchPressed_1** flag will be set to true when switch 1 is pressed and false when it is released. the program will not increment the value of Num when the switch is pressed, but will continue to count from the current value of Num when it is released.

In practice, the 6-pin key switch is not very sure of the pin path when pressed It took some time to figure out the principle of the switch The switch uses a 10k ohm pull-up resistor to connect to the vcc and Arduino pins, and the other end is grounded, so that it outputs a low level to the Arduino pins when pressed.