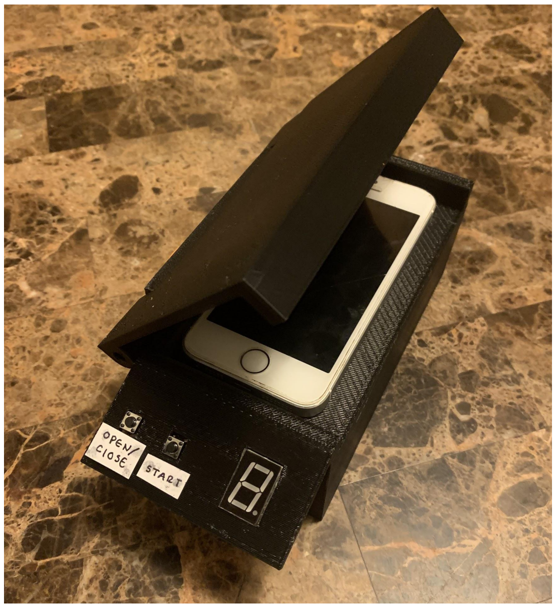
Cellphone Casket Countdown Timer

By Juan Carrillo

Description of the product:

It seems like working from home will become a new norm. Many companies have stated that they will provide the option to work from home to their employees even when the pandemic is over. Living in a generation that is so dependent on technology, it seems as if many of us can't function without our devices. This makes devices, such as cellphones, a constant temptation to steer us away from being as productive in a work from home environment.

Online learning posed several challenges for many of us. Personally, one of them was trying to be productive and staying focused when I always had my phone sitting on my desk. It was very easy to put off my homework and browse through my phone. That is where the idea for a cellphone casket came about. I remembered that one strategy I read online was to put your phone away for 2 hours, and then use it for 10-15 mins. I thought this would be a whole lot easier if I had a box to put my phone away and have a digital display of how much time has elapsed rather than simply trying to ignore my phone. Once a certain amount of time has passed, then the box automatically opens, and I would be able to take a break. Therefore, I designed my system to store my cellphone and have a countdown timer display.



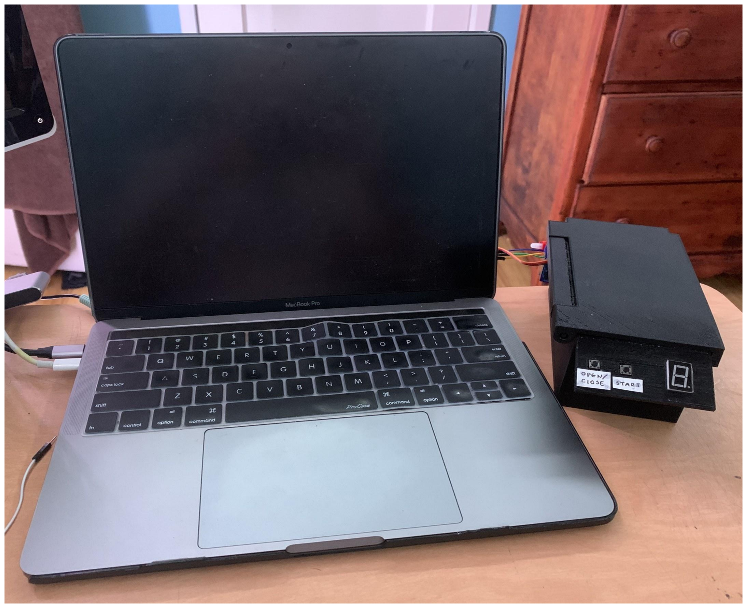
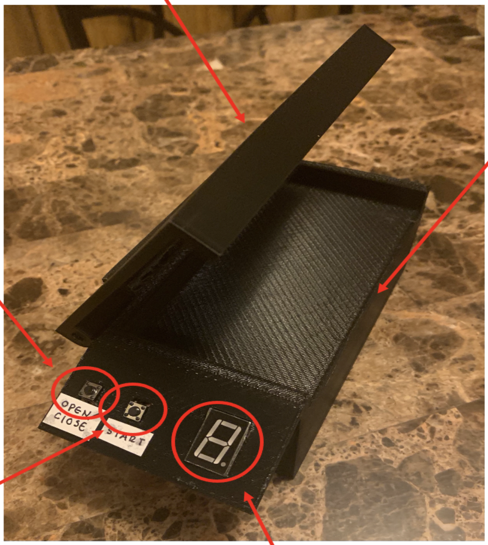
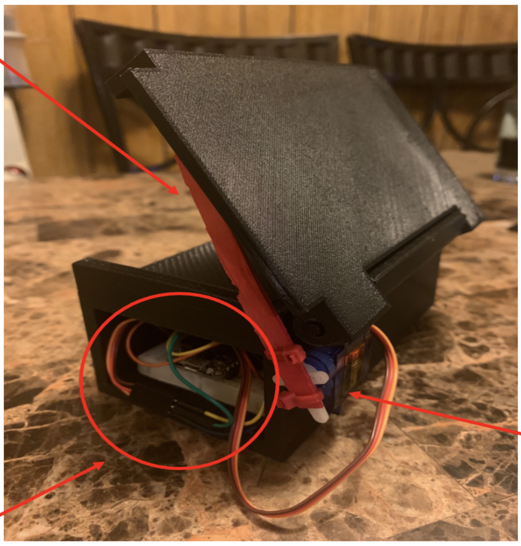


Figure 1: My cellphone casket I designed and 3d printer holding my cellphone (left). The right picture shows where the casket would do in my desk so that I can be productive and have the willpower to not open it.

Electromechanical Details:

<u>Control box:</u> I designed the casket using Fusion 360. The casket is composed of the casket lid, base, hinge bar, and servo motor arm. All pieces were 3d printed. I designed the casket so that the backside has an opening to house the breadboard, micro, and wires. There is a small opening in the front of the box so that the wires can come out and attach to the 2 buttons and 7 segment digital display. The front of the casket has an interface place that holds the 2 buttons and 7 segment digital display. This was done taking into account Human Computer Interaction principles. Making the device user friendly, one button open and closes the lid, the other button starts the countdown timer, and the 7 segment digital display displays how much time is left. The servo motor is attached to the backside of the casket, and the servo motor arm lifts and lowers the casket lid. The USB port is easily accessible through the back housing and the power supply can easily be hooked up to the breadboard.





Circuit:

There were three main elements: (1) the servo motor, (2) two push buttons, (3) and 7 segment digital display.

1) <u>Servo motor:</u> I had a microservo 99 SG90 motor from an arduino starter kit that I gave to my brother last Christmas (the kit was still brand new haha). Since I wanted the servo motor arm to simply open and close the lid, a servo motor was more applicable since I was able to control the position to where it moved. With the Pololu DC motor that we had in our kit, it was almost impossible to control the position since even at it lowest speed, it was still at around 150 rpm. I only wanted the motor to turn 75 degrees (from the

horizontal). For this, the servo motor was perfect. The motor is powered with the 5v power supply.

2) Two push buttons: I used 2 normal push buttons. One was to open and close the lid, and another one to start the countdown timer. The start button only works if the lid is closed. Both buttons work with attach interrupts and a debounce threshold. A 10k pulldown resistor is used for each button.
3) 7 segment digital display: I used a single 7 segment digital display. I tried to use a 4 digit 7 segment digital display (to display minutes and seconds), but it uses so many pins, that I did not have enough on the micro. There is a backpack that allows you to only use 3 pins, but the part would not get here on time (bad scheduling on my part). However, a single digit gets the job done. I used a common anode set up, so that meant that each pin (A-G on the digital display) got a 220 Ohm resistor. I set it up to work as a countdown timer from 10 to 0. For demonstration purposes, I did seconds instead of minutes.

In addition, I used the 5V power supply from out kit to power the servo. The micro is powered by the USB of the computer, but that is okay because the point of the casket is to be placed next to your workstation. The 5V power supply can also be replaced by 4 AAA batteries as I show in my Fritzing diagram (4.8V total).

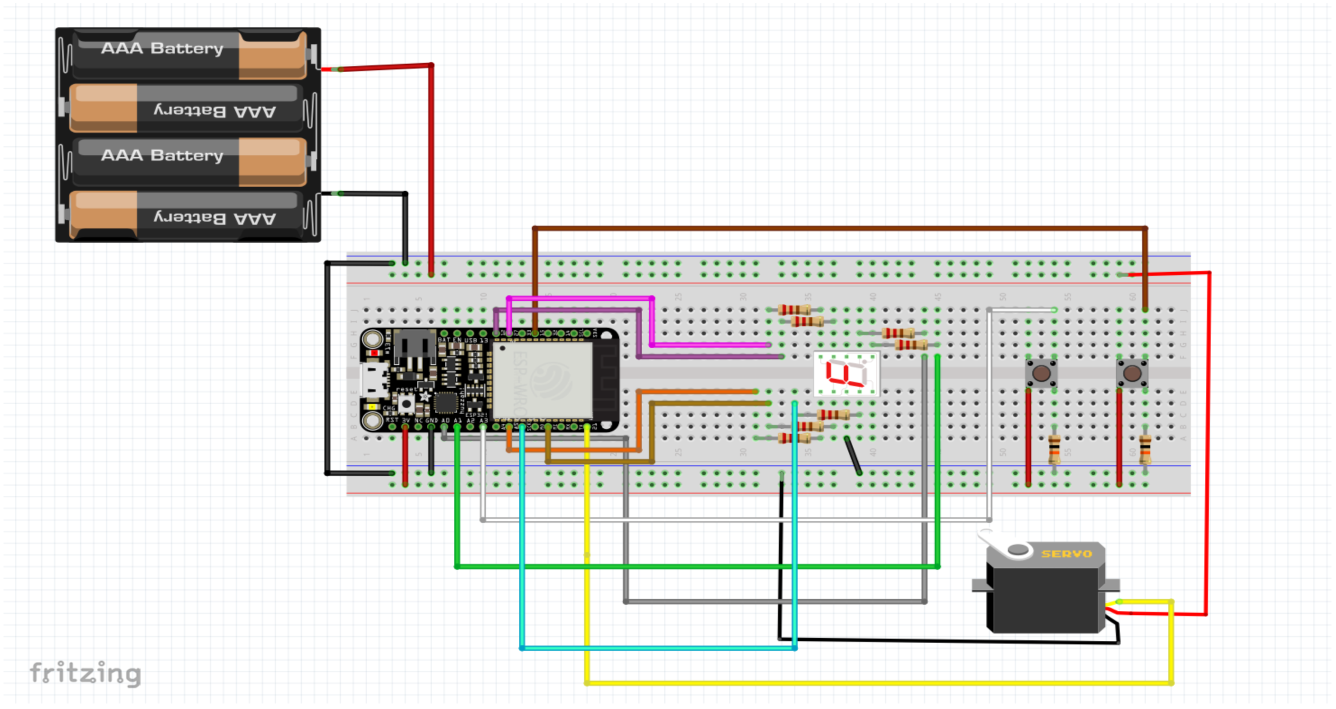


Figure 3: Fritzing Diagram of all the components of Cellphone Casket

Finite State Machine:

The behavior is not extremely complicated. There are 3 different states: (1) lid closed, not counting, (2) lid open, (3) lid closed, counting. By "counting" and "not counting," I am referring to whether the countdown timer is activated or not. Button1 corresponds to the button that is in charge of opening and closing the lid. Button 2 corresponds to the button that starts the countdown timer. In the service of the events that correspond to "close lid" or "open lid," this is the servo motor turning 75 degrees to open, and turning a -75 degrees to close (end up back at starting position). The motor is only on while it turns those degrees, and then it automatically shuts off. All this is demonstrated in the video.

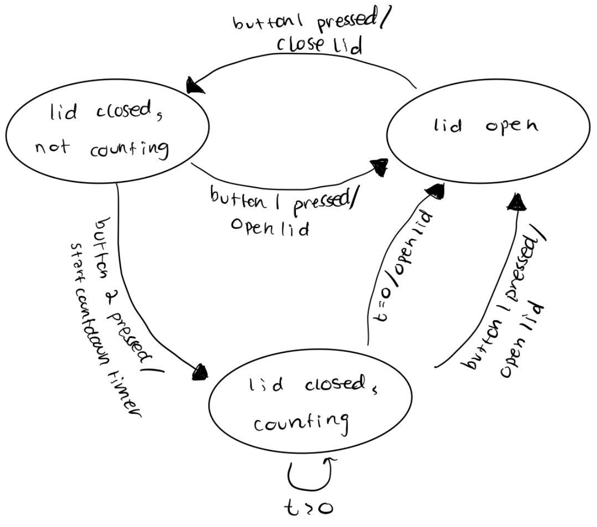

button1 pressed/ close lid
lid closed, not counting
lid open
button 1 pressed/ open lid
button 2 pressed/ start countdown timer
t=0/open lid
button 1 pressed/ open lid
lid closed, counting
t>0

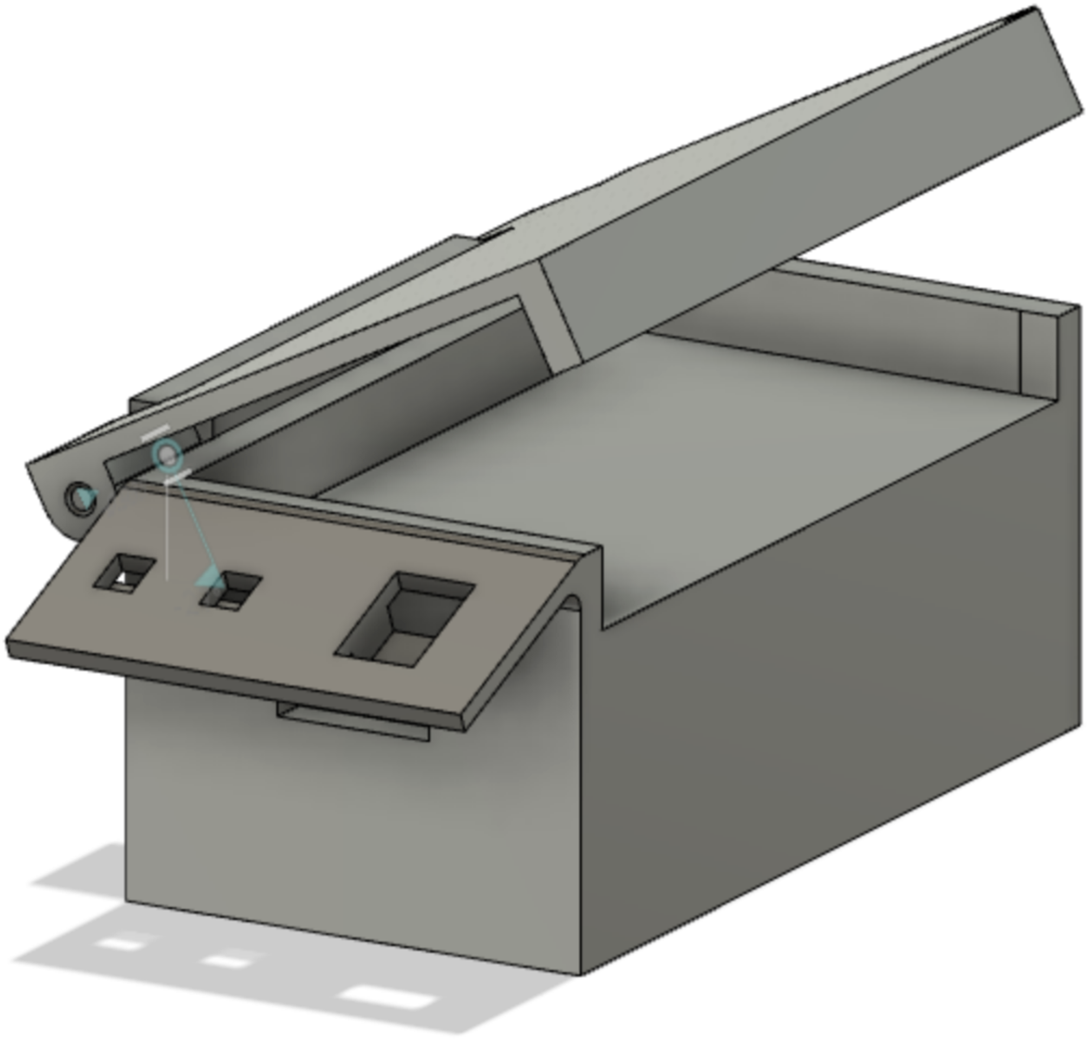
Figure 4: Finite State Machine Diagram. It consists of 3 states.

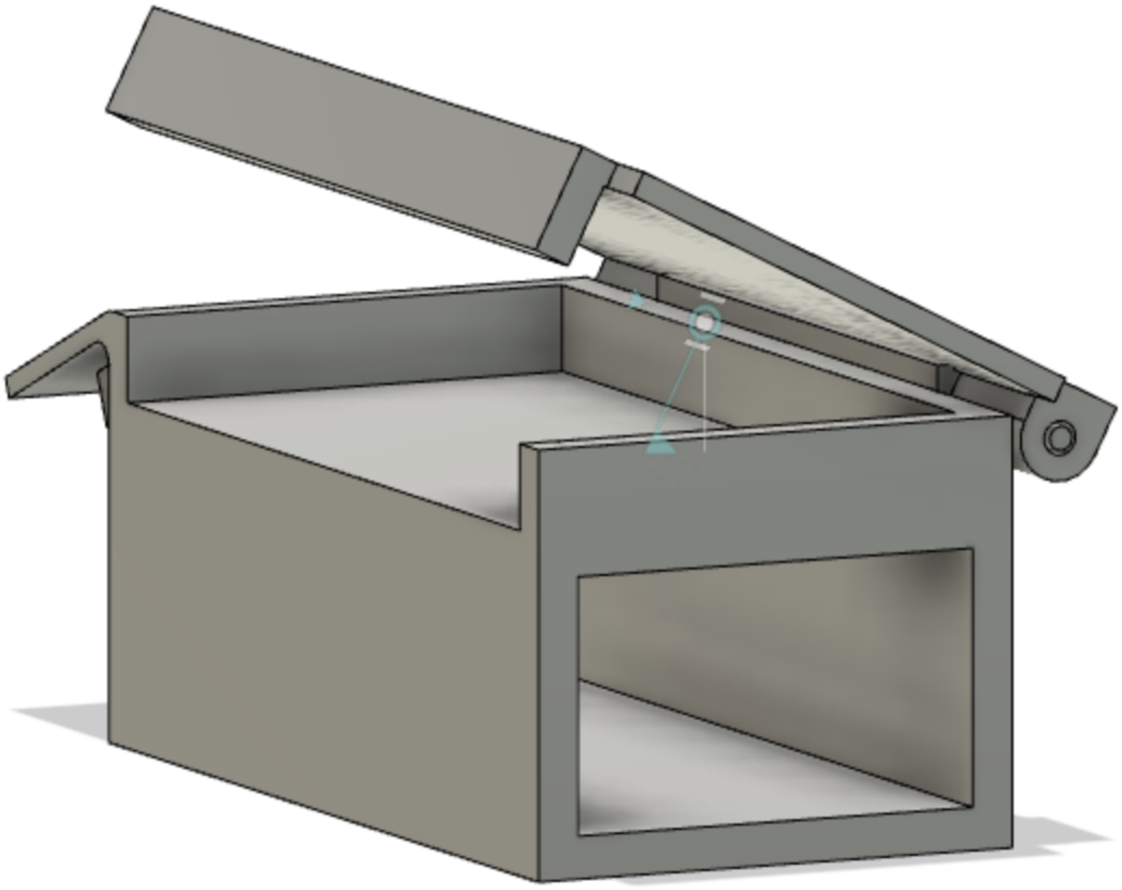
I did not provide any graph because my machine is based on user input. I do not have any sensors that are recording and changing over time. The only thing is my countdown timer. But that also depends on the user when they press the buttons.

My complete Arduino IDE code is provided below, along with screenshots of my Fusion 360 assembly.

Conclusion and Future Work:

I definitely had a lot of fun building this machine. There are many things that you have to take into consideration when designing and building a mechatronic system. In the future, I definitely want to implement the 4 digit 7 segment display or incorporate an LCD display to better display the countdown timer. It would also be cool to implement a speaker that alarms you when the time is up (since right now I only have the lid opening by itself). I think that a lot of work can be put into the design of the case and maybe implementing a locking mechanism that completely locks the lid once it closes. Again, I had a lot of fun, and hopefully you enjoyed reading about my project.





```
#include <Servo_ESP32.h>
#define btn_pin1 39
#define btn_pin2 15
// State machine variables
boolean state = 0;
volatile boolean buttonPressEvent = false;
volatile boolean state_btn = false;
volatile int buttonTimer = 0;
volatile int buttonTimer_last = 0;

// Thresholds
#define DEBOUNCE 200

const int a = 26;
const int b = 25;
const int c = 16;
const int d = 19;
const int e = 4;
const int f = 12;
const int g = 27;
int p = 13;

int startStopReset = 13;

volatile int interruptCounter;
int totalInterruptCounter;

hw_timer_t * timer = NULL;
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;

void IRAM_ATTR onTimer() {
  portENTER_CRITICAL_ISR(&timerMux);
  portEXIT_CRITICAL_ISR(&timerMux);
  if (buttonPressEvent == false && state_btn == true){
    interruptCounter ++;
  }
  if (totalInterruptCounter == 1){
    nine();
  }
   else if (totalInterruptCounter == 2){
    eight();
  }
   else if (totalInterruptCounter == 3){
    seven();
   }
```

```
     else if (totalInterruptCounter == 4){
     six();
    }
     else if (totalInterruptCounter == 5){
     five();
    }
        else if (totalInterruptCounter == 6){
     four();
    }
        else if (totalInterruptCounter == 7){
     three();
    }
        else if (totalInterruptCounter == 8){
     two();
    }
        else if (totalInterruptCounter == 9){
     one();
    }
    else if (totalInterruptCounter == 10){
     open_lid();
     totalInterruptCounter = 0;
     interruptCounter =0;
     state_btn = false;
     zero();
    }

}

static const int servoPin = 21; //printed G14 on the board

Servo_ESP32 servo1;

void setup(){

  pinMode(a, OUTPUT);
  pinMode(b, OUTPUT);
  pinMode(c, OUTPUT);
  pinMode(d, OUTPUT);
  pinMode(e, OUTPUT);
  pinMode(f, OUTPUT);
  pinMode(g, OUTPUT);
  pinMode(p, OUTPUT);
  pinMode(startStopReset, INPUT);
  digitalWrite(startStopReset, HIGH);
  Serial.begin(115200);

  servo1.attach(servoPin);

  pinMode(btn_pin1,INPUT);
  attachInterrupt(digitalPinToInterrupt(btn_pin1),buttonIsPressed1,RISING);

  pinMode(btn_pin2,INPUT);
  attachInterrupt(digitalPinToInterrupt(btn_pin2),buttonIsPressed2,RISING);

  timer = timerBegin(0, 80, true);
  timerAttachInterrupt(timer, &onTimer, true);
  timerAlarmWrite(timer, 1000000, true);
  timerAlarmEnable(timer);
}

void loop(){

  if (interruptCounter > 0) {

    portENTER_CRITICAL(&timerMux);
    interruptCounter--;
    portEXIT_CRITICAL(&timerMux);

    totalInterruptCounter++;
  }
}

void buttonIsPressed1(){
  zero();
  buttonTimer = millis();
  state_btn =! state_btn;
  Serial.println(buttonPressEvent);
}

void buttonIsPressed2(){
  if (buttonPressEvent == false){
     servo1.write(75);
  }
  else if (buttonPressEvent == true) {
    servo1.write(0);
  }
  zero();
  buttonTimer = millis();
  buttonPressEvent =! buttonPressEvent;
  Serial.println(buttonPressEvent);
}

void dispDec(int x)
{
  digitalWrite(p, LOW);
}

void clearLEDs()
{
  digitalWrite(a, LOW);
  digitalWrite(b, LOW);
  digitalWrite(c, LOW);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, LOW);
  digitalWrite(p, LOW);
}

void open_lid(){
  servo1.write(75);
  buttonPressEvent =! buttonPressEvent;
}

void zero()
{
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, HIGH);
  digitalWrite(g, LOW);
}

void one()
{
  digitalWrite(a, LOW);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, LOW);
```

```
}

void two()
{
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, LOW);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, LOW);
  digitalWrite(g, HIGH);
}

void three()
{
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, HIGH);
}

void four()
{
  digitalWrite(a, LOW);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
}

void five()
{
  digitalWrite(a, HIGH);
  digitalWrite(b, LOW);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, LOW);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
}
```

```
void six()
{
  digitalWrite(a, HIGH);
  digitalWrite(b, LOW);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
}

void seven()
{
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, LOW);
}

void eight()
{
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
}

void nine()
{
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, LOW);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
}
```