/\*Handy code blocks\*/

//Variables for LED/ESP-Now

const int ledPin = 2;         // Define the GPIO pin connected to the LED

bool receivedValue = false;   // Variable to store the received boolean value

bool lightFunctionCalled = false;                 // Internal flag to track if timing has started

//Variables for Switch case

unsigned long previousTime = 0;         // Stores the time when the LED was turned on

unsigned long newTime = 0;        //Time updates from millis() every loop

int state = 0;                       // Tracks the current state in the switch-case

//$$$$$$$$$$$$$$$Debugging variables

int loopDebug ; //a contrivance to show loop is working but only write 5 times, not fill the whole page

int inByte = 0; //the incoming keyboard bytes for keyboard debugging

bool deBugSensors = false; // true when deBugByte = 'a'

bool deBugReset = false; // true when deBugByte = 'b'

bool slaveOnhold; // gets read within sensor function, only tripped if sensors < theshold && slaveOnhold = false

bool slaveSprung; //\*\*\*\* read where, or signaling Master--same for both slaves so don't need ABCD

long stateMachineTimeOutmillis; // for timed transitions out of machine states

//\*\*\*\*\*

//Switch case for LED and bools

void handleReceivedValue() {

  switch (state) {

     case 0:   // Case 0: monitor bool

      if (lightFunctionCalled) {                  // If the handleReceivedValue has been true

      previousTime = millis();

      state = 1;

    }

    break;

    case 1:   // Case 1: Turn on LED and start timing

        digitalWrite(ledPin, HIGH);      // Turn on the LED

      if (newTime) - previousTime >= 2000; { // Check if 2000 ms have passed

        state = 2;                        // Move to the next state

     }

      break;

    case 2:   // Case 2: Turn off LED and reset the bool receivedValue

      digitalWrite(ledPin, LOW);         // Turn off the LED

      lightFunctionCalled = false;                   // Reset the LOCAL boolean

      state = 0;                         // Reset state for next cycle

      break;

  }

}

void loop() {

  // Main loop was empty; action is handled in the callback

   newTime = millis();            // Save the current time each loop

 //onDataReceived();

  handleReceivedValue();   //call this function every loop to turn on LED if recieved value becomes true

}

//\*\*\*\*\*Include this to serialPrint the 1st 5 loops to show it is working but not obscure serial //monitor.

 //Debug to check loop is working

  if (loopDebug > 0 )

  {

    Serial.print(F( "Looping..." ));

    Serial.println(loopDebug);

  }

  loopDebug = --loopDebug ;                      //decrement from 5 , stop at 0

  //  Serial.println (loopDebug);

  if (Serial.available() > 0) {

    int inByte = Serial.read();                   //for keyboard debugging

Serial.println(inByte);

//\*\*\*\*\* Include this in loop for serialMonitor debugging. Change code to suit test variables.

Serial.println(inByte);

  switch (inByte) {                             //for keyboardd debugging

    case 'a':

      // make the servoA cycle

      Serial.println(F("I just typed 'a' to call the servo function"));

     // if (!hasRun) {

        tubeAfull = true;

        DOOR\_state = DOOR\_STATE\_INITIALISE;

     // }

      //trick a transition to state 1

      break;

    case 'b':

      // make the servoA cycle

      Serial.println(F("I just typed 'b' to call the servo function"));

   //   if (!hasRun) {

       tubeBfull = true;

        DOORB\_state = DOORB\_STATE\_INITIALISE;

      //}

      break;

    case 'c':

    //  Serial.println(F("I just typed 'c' to clear the hasRun"));

      //hasRun = false;

      // State = 1;                                                                    //trick a transition to state 1

      break;

    case 'd':

      Serial.println(F("I just typed 'd' to "));

      //

      break;

    case 'e':

      Serial.println(F("I just typed 'e' to  reset fake news debugs"));    //    Restore variables

      //        deBugBattery = false ;

      deBugReset = false;

      //        deBugBattery = false;

      //      deBugFakeAwaiting = false;

      break;

    case 'z':

      Serial.println(F("I just typed 'z' to stop the serial"));

      while (1) {}        //Debugging reset, needs a manual reset and close/reopen serial minitor after this STOP method

      break;

  }

  }                          //End switch