Desk Stand Up Reminder

Navarro, Rod Geryk C.

CPE160P - 4– A1

Introduction

•Prolonged periods of sitting are associated with various health issues, highlighting the need for regular movement in daily routines.

•The Desk Stand Up Reminder project utilizes an Arduino Uno microcontroller and integrates components like a 4-bit seven-segment display, an LCD I2C 1602, LEDs, a buzzer, and an ultrasonic sensor to encourage users to stand and move at regular intervals.

•Following the 20-8-2 rule, the system promotes a balanced work habit of sitting for 20 minutes, standing for 8 minutes, and moving for 2 minutes in each 30-minute cycle, with users able to adjust their sitting and standing times according to preference.

•The ultrasonic sensor enables hands-free activation or reset of the system, and the project illustrates the practical application of embedded system skills while fostering a healthier, more active lifestyle.

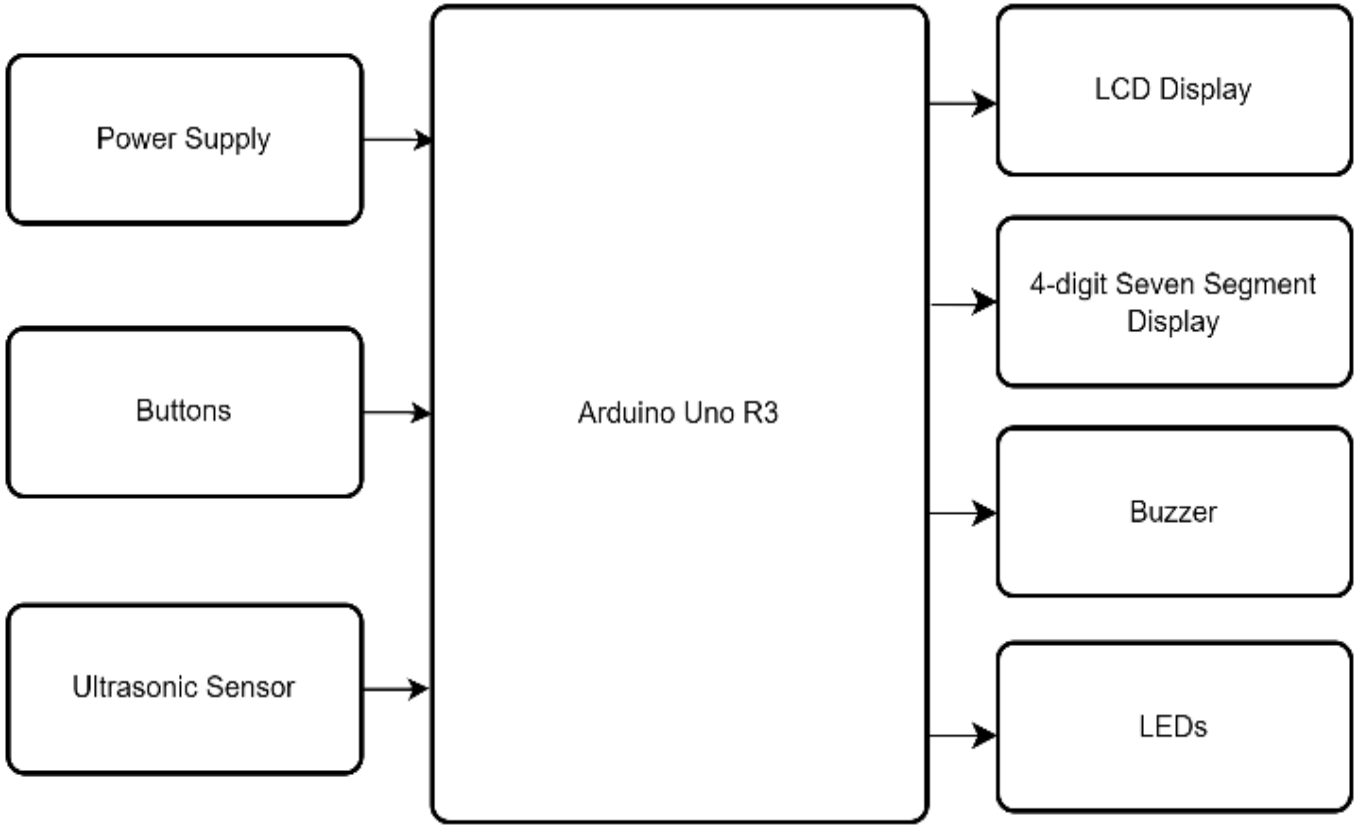
OBJECTIVES

- To be able to demonstrate and create a stand-up reminder using Arduino R3.
- To be able to apply embedded system in reminding the user when to stand-up if the user sits for a long period of time using LCD, and 4- bit seven segment display, as well as the use buttons, ultrasonic sensor and a buzzer.
- To be able to effectively remind the user when the time to stand up especially if the user has the habit of sitting for a long period of time or has a work that allows the user to sit for a long time.
- To be able to use the skills learned and the functionality of at least five of the seven embedded system and design experiments.

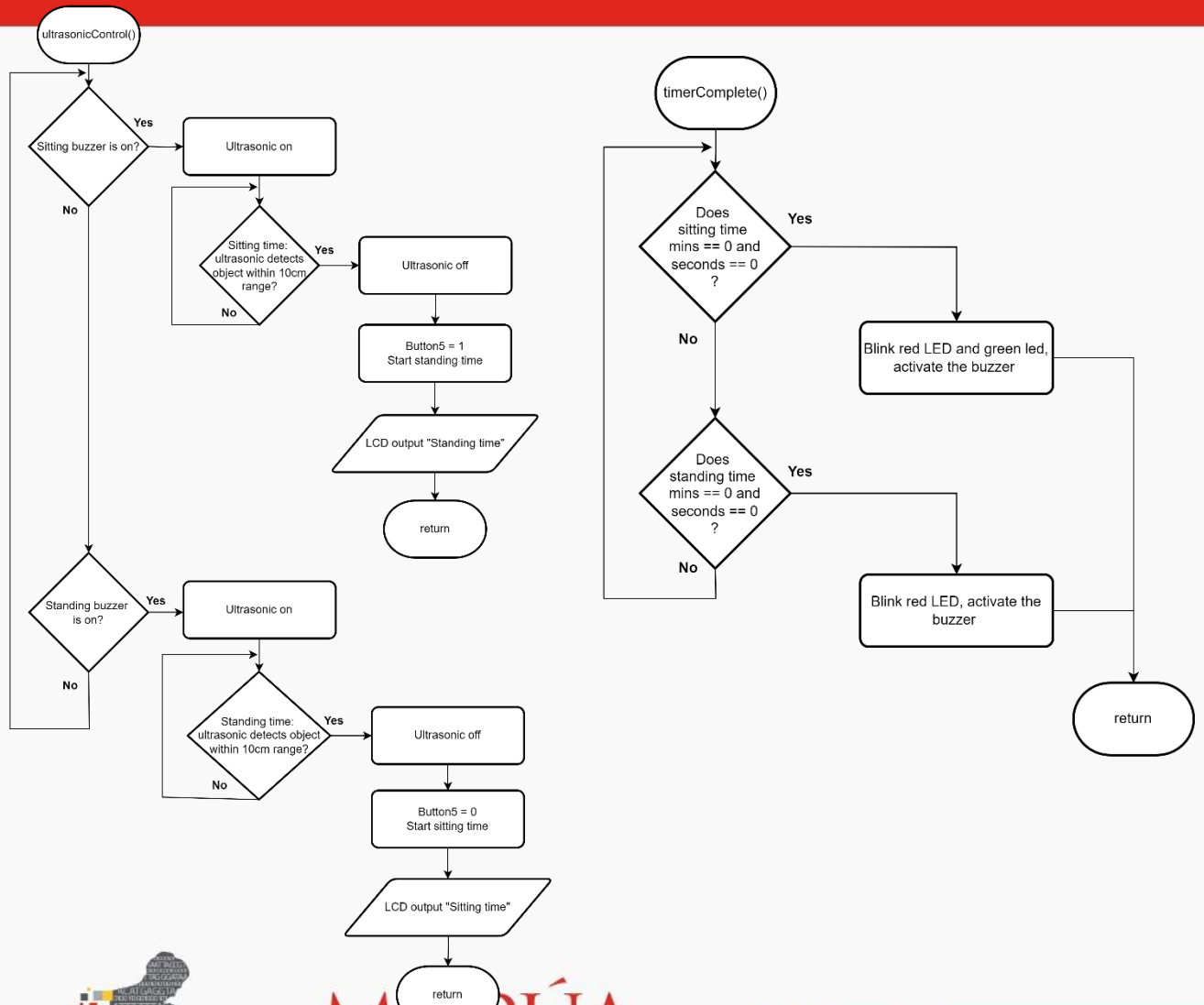
MATERIALS

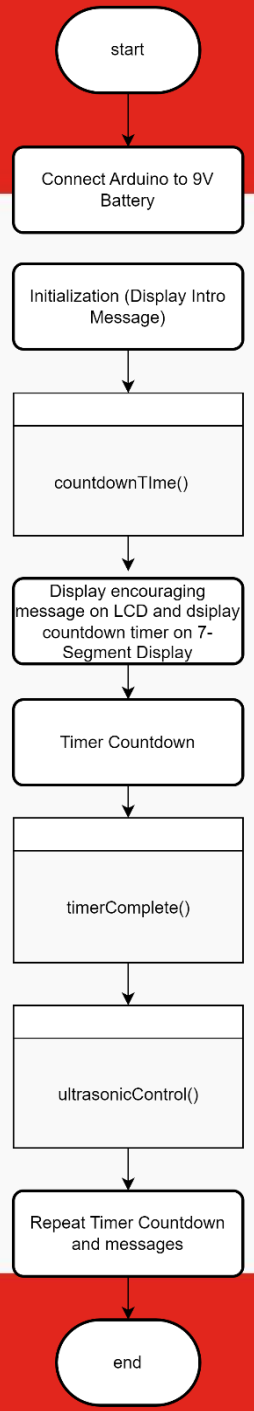
- 1x Arduino kit
- 1x Arduino R3
- 5x Switch/buttons w/ resistors
- 1x LCD (16x2)
- 1x 4-digit seven segment display w/ resistors
- 2x LEDs w/ resistors
- 1x Ultrasonic sensor
- 1x Buzzer w/resistor

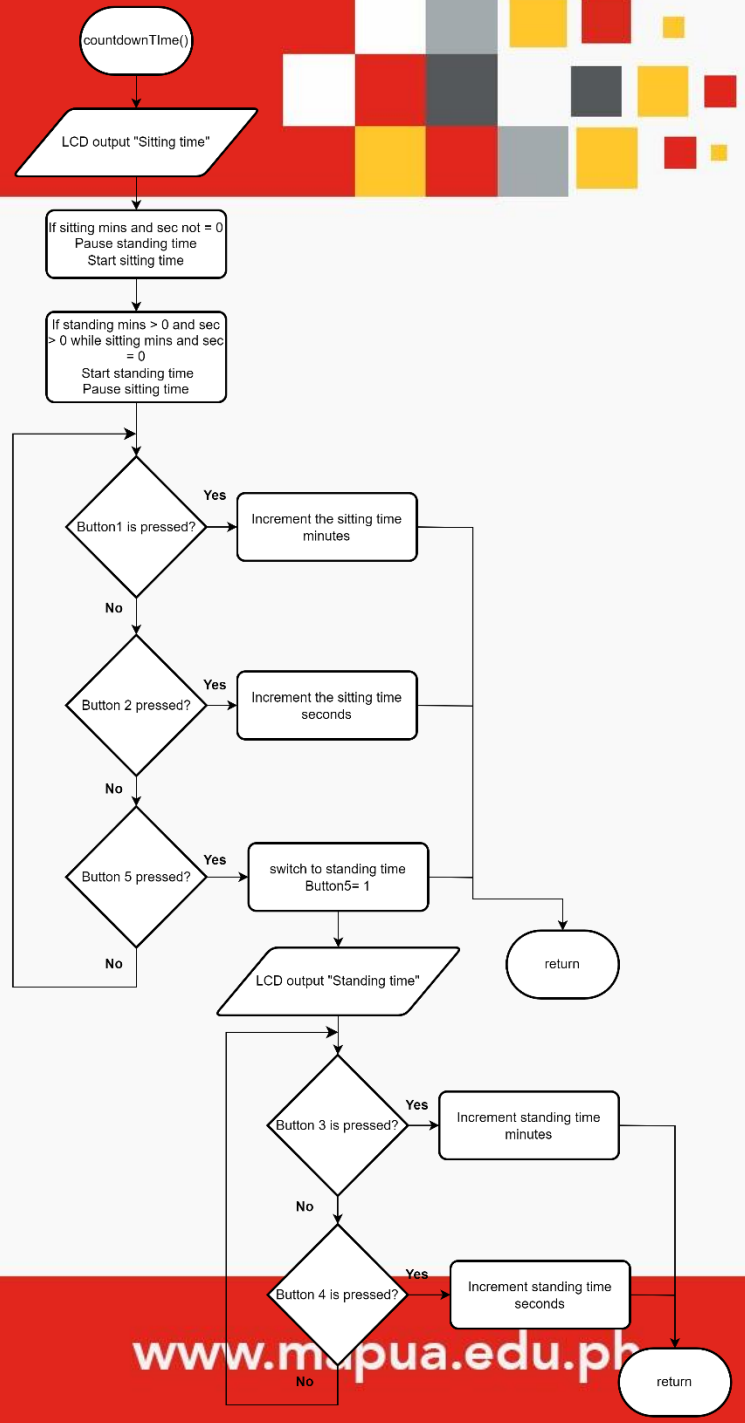
Block Diagram



Flowchart







MAPÚA UNIVERSITY

www.mapua.edu.ph

PROGRAM LISTING

```ino
#include <LiquidCrystal_I2C.h> // Include library for I2C LCD control

// Define pins for 7-segment display
const int sPins[7] = {A3, A2, A1, 12, 8, 9, 10}; // Segment pins
const int dPins[4] = {4, 5, 6, 7}; // Digit pins
const int dppin = 1; // Unused digit pin
const int buzzAlarm = 11; // Buzzer pin
const int ledGreen = 13; // Green LED pin
const int triggerPin = 3; // Ultrasonic sensor trigger pin
const int echoPin = 2; // Ultrasonic sensor echo pin

const int switch1Pin = A0; // Pin for button input
int sittingMinutes = 19; // Default sitting timer duration
int standingMinutes = 9; // Default standing timer duration
int maxMinutes = 30; // Maximum allowed minutes for timers
bool settingSitting = true; // Flag to track current timer mode (sitting or standing)
bool adjustMode = true; // Flag for adjustment mode

// Variables for timing and message display
unsigned long previousMillis = 0; // Store last time update
unsigned long messageMillis = 0; // Store last message time
const long interval = 1000; // 1 second interval for countdown
const long messageInterval = 5000; // 5 seconds for each motivational message on LCD
int minutes = sittingMinutes; // Initialize current minutes
int seconds = 0; // Initialize seconds
bool timerRunning = true; // Flag to control timer state (running or paused)
int messageIndex = 0; // Index for motivational messages
int cm = 0; // Variable for ultrasonic distance measurement

// Byte array for displaying digits on the 7-segment display
byte digitCodes[10] = {
  B00111111, B00000110, B01011011, B01001111,
  B01100110, B01101101, B01111101, B00000111,
  B01111111, B01101111
```

```ino
// Create an instance of the LCD display
LiquidCrystal_I2C dis(0x27, 16, 2); // I2C address, columns, rows

// Motivational messages for sitting and standing
const char* sittingMessages[] = {
  " Focus on Work!  ", " Keep Grinding!  ", "  Stay Sharp!!  ",
  "You're on Track!", "Reach your Goal"
};

const char* standingMessages[] = {
  "Stand & Stretch!", "Move your Body!!", " Shake it out!!  ",
  "Feel Refreshed!!", " Time to Stand!  "
};

void setup() {
  Serial.begin(9600); // Start serial communication for debugging

  // Initialize pin modes
  pinMode(buzzAlarm, OUTPUT); // Set buzzer pin as output
  pinMode(ledGreen, OUTPUT); // Set green LED pin as output
  pinMode(triggerPin, OUTPUT); // Set ultrasonic trigger pin as output
  pinMode(echoPin, INPUT); // Set ultrasonic echo pin as input

  // Set segment pins as output
  for (int i = 0; i < 7; i++) {
    pinMode(sPins[i], OUTPUT);
  }

  // Set digit pins as output and turn them off initially
  for (int i = 0; i < 4; i++) {
    pinMode(dPins[i], OUTPUT);
    digitalWrite(dPins[i], HIGH);
  }
```

PROGRAM LISTING

```
}

pinMode(dppin, OUTPUT); // Set unused digit pin as output
displayTime(minutes, seconds); // Display initial time

// Initialize and display startup messages on the LCD
dis.init();
dis.backlight();
dis.clear();
dis.setCursor(0, 0);
dis.print("StandUp Reminder");
dis.setCursor(3, 1);
dis.print("Good Day!");

// Display a series of dots as a loading animation
for (int a = 12; a <= 15; a++) {
  dis.setCursor(a, 1);
  dis.print(".");
  delay(1500);
}
dis.clear(); // Clear the display after the loading animation


void loop() {
// Read analog value from the button
int buttonValue = analogRead(switch1Pin);
Serial.print("Analog value: ");
Serial.print(buttonValue); // Print button value for debugging

// Read the digital state of the button
int allButtons = digitalRead(switch1Pin);

// Determine button press actions based on the analog value
if (buttonValue < 100) {
```

```
if (buttonValue < 100) {
  allButtons = LOW; // No button pressed
} else if (buttonValue < 150) {
  incrementTimer(true); // Increment sitting timer
  delay(500); // Debounce delay
} else if (buttonValue < 250) {
  decrementTimer(true); // Decrement sitting timer
  delay(500); // Debounce delay
} else if (buttonValue < 350) {
  incrementTimer(false); // Increment standing timer
  delay(500); // Debounce delay
} else if (buttonValue < 850) {
  decrementTimer(false); // Decrement standing timer
  delay(500); // Debounce delay
} else {
  switchTimerMode(); // Switch between sitting and standing timer
  delay(500); // Debounce delay
}

// Timer logic
if (timerRunning) {
  unsigned long currentMillis = millis(); // Get current time in milliseconds
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis; // Update previousMillis
    if (seconds == 0) { // Check if seconds are zero
      if (minutes > 0) {
        minutes--; // Decrease minutes
        seconds = 59; // Reset seconds to 59
      } else {
        completeTimer(); // Timer completed
      }
    } else {
      seconds--; // Decrease seconds
    }
```

PROGRAM LISTING

```
    // Display motivational message at specified intervals
    if (currentMillis - messageMillis >= messageInterval) {
      messageMillis = currentMillis; // Update messageMillis
      displayMotivationalMessage(); // Show a motivational message
    }
  }

  displayTime(minutes, seconds); // Update the displayed time
}

void incrementTimer(bool sitting) {
  // Increment the timer based on whether it's the sitting or standing timer
  if (sitting) {
    if (sittingMinutes < maxMinutes) sittingMinutes++; // Increment sitting minutes
    if (settingSitting) minutes = sittingMinutes; // Update displayed minutes if sitting timer is active
  } else {
    if (standingMinutes < maxMinutes) standingMinutes++; // Increment standing minutes
    if (!settingSitting) minutes = standingMinutes; // Update displayed minutes if standing timer is active
  }
}

void decrementTimer(bool sitting) {
  // Decrement the timer based on whether it's the sitting or standing timer
  if (sitting) {
    if (sittingMinutes > 0) sittingMinutes--; // Decrement sitting minutes
    if (settingSitting) minutes = sittingMinutes; // Update displayed minutes if sitting timer is active
  } else {
    if (standingMinutes > 0) standingMinutes--; // Decrement standing minutes
    if (!settingSitting) minutes = standingMinutes; // Update displayed minutes if standing timer is active
  }
}

void switchTimerMode() {
  // Switch between sitting and standing timer modes
```

```
void switchTimerMode() {
  // Switch between sitting and standing timer modes
  settingSitting = !settingSitting; // Toggle the mode
  timerRunning = true; // Resume the timer

  if (settingSitting) {
    minutes = sittingMinutes; // Set minutes to sitting timer
    Serial.println("Switched to Sitting Timer");
    dis.setCursor(2, 1);
    dis.print("Sitting Time "); // Display current mode on LCD
    seconds = 59; // Reset seconds to 59
  } else {
    minutes = standingMinutes; // Set minutes to standing timer
    Serial.println("Switched to Standing Timer");
    dis.setCursor(2, 1);
    dis.print("Stand Up Time"); // Display current mode on LCD
    seconds = 59; // Reset seconds to 59
  }
}

void displayMotivationalMessage() {
  // Display motivational messages based on current timer mode
  dis.setCursor(0, 0);
  if (settingSitting) {
    dis.print(sittingMessages[messageIndex]); // Show sitting message
  } else {
    dis.print(standingMessages[messageIndex]); // Show standing message
  }
  messageIndex = (messageIndex + 1) % 5; // Cycle through messages
}

void displayTime(int mins, int secs) {
  // Convert minutes and seconds to digits and display on 7-segment
  int digits[4] = {mins / 10, mins % 10, secs / 10, secs % 10}; // Split time into digits
```

PROGRAM LISTING

```
89
90  void displayMotivationalMessage() {
91    // Display motivational messages based on current timer mode
92    dis.setCursor(0, 0);
93    if (settingSitting) {
94      dis.print(sittingMessages[messageIndex]); // Show sitting message
95    } else {
96      dis.print(standingMessages[messageIndex]); // Show standing message
97    }
98    messageIndex = (messageIndex + 1) % 5; // Cycle through messages
99  }
00
01  void displayTime(int mins, int secs) {
02    // Convert minutes and seconds to digits and display on 7-segment
03    int digits[4] = {mins / 10, mins % 10, secs / 10, secs % 10}; // Split time into digits
04
05    for (int i = 0; i < 4; i++) {
06      digitalWrite(dPins[i], LOW); // Activate digit pin
07      setSegments(digitCodes[digits[i]]); // Display corresponding segment code
08      delay(5); // Short delay for visibility
09      digitalWrite(dPins[i], HIGH); // Deactivate digit pin
10    }
11  }
12
13  void setSegments(byte segments) {
14    // Set segments of the 7-segment display
15
```

Prototype Demonstration

THANK YOU!