SPRINT – 1

Date	29 October 2022
Team ID	PNT2022TMID52309
Project Name	Personal Assistance For Seniors Who Are Self-Reliant

TASK:-

To simulate Arduino using python code.

DESCRIPTION:

* For the construction of IoT devices, we used the IoT Watson platform.

*Node-RED is used in the development of the web application to collect user input regarding prescription medications.

*The data that was gathered was stored using cloudant DB.

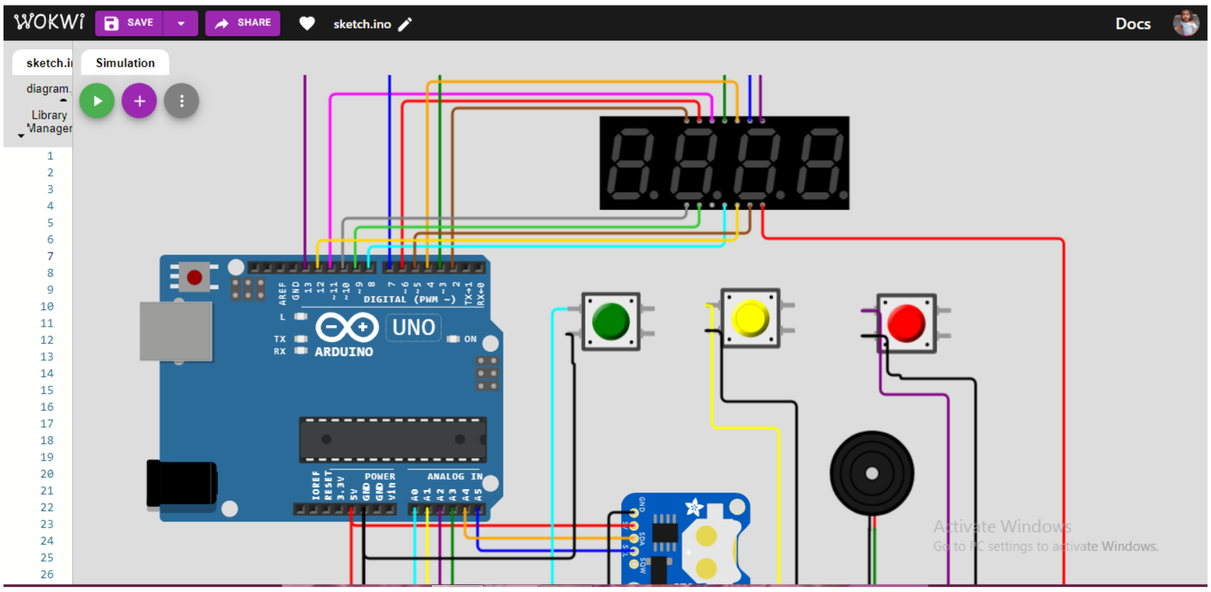
*The created IoT device will receive the medication details from the web application.

*After receiving the information, the IoT gadget uses TTS to remind the user to take their medication.

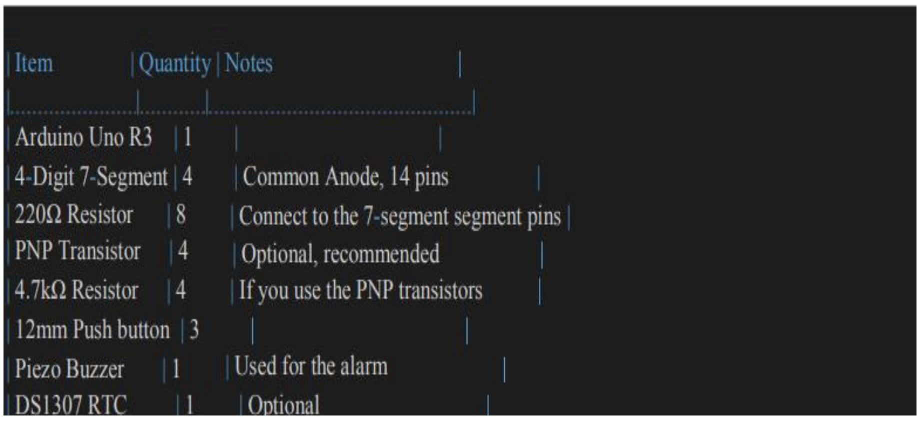
*The users will be informed of the medical information via voice commands using the IBM platform's TTS (Text to Speech) capability

*The screenshots that show how to simulate an Arduino board in the Workwi simulator are provided below.

*Establish the appropriate link in the workwi simulator to set an alert.



EQUIPMENTS REQUIRED:



Item	Quantity	Notes
Arduino Uno R3	1	
4-Digit 7-Segment	4	Common Anode, 14 pins
220Ω Resistor	8	Connect to the 7-segment segment pins
PNP Transistor	4	Optional, recommended
4.7kΩ Resistor	4	If you use the PNP transistors
12mm Push button	3	
Piezo Buzzer	1	Used for the alarm
DS1307 RTC	1	Optional

CODING PART:

```
 /**
Arduino Digital Alarm Clock
*/
#include <SevSeg.h>
#include "Button.h"
#include "AlarmTone.h"
#include "Clock.h"
#include "config.h"
const int COLON_PIN = 13;
const int SPEAKER_PIN = A3;
Button hourButton(A0);
Button minuteButton(A1);
Button alarmButton(A2);
AlarmTone alarmTone;
Clock clock;
SevSeg sevseg;
enum DisplayState {
DisplayClock,
DisplayAlarmStatus,
DisplayAlarmTime,
DisplayAlarmActive,
DisplaySnooze,
};
DisplayState displayState = DisplayClock;
long lastStateChange = 0;
void changeDisplayState(DisplayState newValue) {
displayState = newValue;
lastStateChange = millis();
}
long millisSinceStateChange() {
return millis() - lastStateChange;
}
void setColon(bool value) {
digitalWrite(COLON_PIN, value ? LOW : HIGH);
}
void displayTime() {
DateTime now = clock.now();
bool blinkState = now.second() % 2 == 0;
sevseg.setNumber(now.hour() * 100 + now.minute());
```

```
setColon(blinkState);
}
void clockState() {
displayTime();
if (alarmButton.read() == Button::RELEASED &&
clock.alarmActive()) {
// Read alarmButton has_changed() to clear its state
alarmButton.has_changed();
changeDisplayState(DisplayAlarmActive);
return;
}
if (hourButton.pressed()) {
clock.incrementHour();
}
if (minuteButton.pressed()) {
clock.incrementMinute();
}
if (alarmButton.pressed()) {
clock.toggleAlarm();
changeDisplayState(DisplayAlarmStatus);
}
```

```
}
void alarmStatusState() {
setColon(false);
sevseg.setChars(clock.alarmEnabled() ? " on" : " off");
if (millisSinceStateChange() >
ALARM_STATUS_DISPLAY_TIME) {
changeDisplayState(clock.alarmEnabled() ?
DisplayAlarmTime : DisplayClock);
return;
}
}
void alarmTimeState() {
DateTime alarm = clock.alarmTime();
sevseg.setNumber(alarm.hour() * 100 + alarm.minute(), -1);
if (millisSinceStateChange() >
ALARM_HOUR_DISPLAY_TIME || alarmButton.pressed())
{
changeDisplayState(DisplayClock);
return;
}
if (hourButton.pressed()) {
clock.incrementAlarmHour();
```

```
lastStateChange = millis();
}
if (minuteButton.pressed()) {
clock.incrementAlarmMinute();
lastStateChange = millis();
}
if (alarmButton.pressed()) {
changeDisplayState(DisplayClock);
}
}
void alarmState() {
displayTime();
if (alarmButton.read() == Button::RELEASED) {
alarmTone.play();
}
if (alarmButton.pressed()) {
alarmTone.stop();
}
if (alarmButton.released()) {
alarmTone.stop();
bool longPress = alarmButton.repeat_count() > 0;
```

```
if (longPress) {
clock.stopAlarm();
changeDisplayState(DisplayClock);
} else {
clock.snooze();
changeDisplayState(DisplaySnooze);
}
}
}
void snoozeState() {
sevseg.setChars("****");
if (millisSinceStateChange() > SNOOZE_DISPLAY_TIME)
{
changeDisplayState(DisplayClock);
return;
}
}
void setup() {
Serial.begin(115200);
clock.begin();
hourButton.begin();
hourButton.set_repeat(500, 200);

minuteButton.begin();

minuteButton.set_repeat(500, 200);

alarmButton.begin();

alarmButton.set_repeat(1000, -1);

alarmTone.begin(SPEAKER_PIN);

pinMode(COLON_PIN, OUTPUT);

byte digits = 4;

byte digitPins[] = {2, 3, 4, 5};

byte segmentPins[] = {6, 7, 8, 9, 10, 11, 12};

bool resistorsOnSegments = false;

bool updateWithDelays = false;

bool leadingZeros = true;

bool disableDecPoint = true;

sevseg.begin(DISPLAY_TYPE, digits, digitPins,
segmentPins, resistorsOnSegments,

updateWithDelays, leadingZeros, disableDecPoint);

sevseg.setBrightness(90);

}

void loop() {

sevseg.refreshDisplay();
```

```
switch (displayState) {
case DisplayClock:
clockState();
break;
case DisplayAlarmStatus:
alarmStatusState();
break;
case DisplayAlarmTime:
alarmTimeState();
break;
case DisplayAlarmActive:
alarmState();
break;
case DisplaySnooze:
snoozeState();
break;
}
}
```

OUTPUT:

* At first it shows the current time.

*Turning on the alarm will set a specific time.

*Press the Minute/Hour buttons to set the time. The alarm is turned on or off by pressing the Alarm button. The words "on" or "off" will appear on the screen to indicate the alarm's status.

* The current alarm time will be shown for a short while after the alarm is enabled. The Minute/Hour buttons can be used to change the alarm time.

* Press the Alarm button once more, or simply wait a few seconds, to complete.

* As soon as we set the alarm, it begins to ring.

RESULT:

Thus, by the end of the sprint-1, we developed the code for our alarm simulation using Arduino- UNO in wokwi simulator…!