SPRINT-1

| Date	29 October 2022
Team ID	PNT2022TMID46387
Project Name	Personal Assistance for Seniors Who Are Self Reliant

TASK:-

To simulate Arduino using python code.

DESCRIPTION: -

❖ We have used **IoT Watson platform** for the creation of IoT device.

❖ The web application is built using **Node-RED** for collecting the medicine details from the users.

❖ We have used the **cloudant DB** for storing the collected data.

❖ The web application will send the medicine details to the created IoT device.

❖ The IoT device on receiving the details, it makes use of TTS to remind the user about the medicine.

❖ By using **TTS** (Text to Speech) service from the IBM platform, the medicinal information will be notified to the users in the form of voice commands.

Following are the screenshots that demonstrate the simulation of Arduino using python code in wokwi simulator

❖ Make the necessary connection in wokwi simulator for alarm reminder.
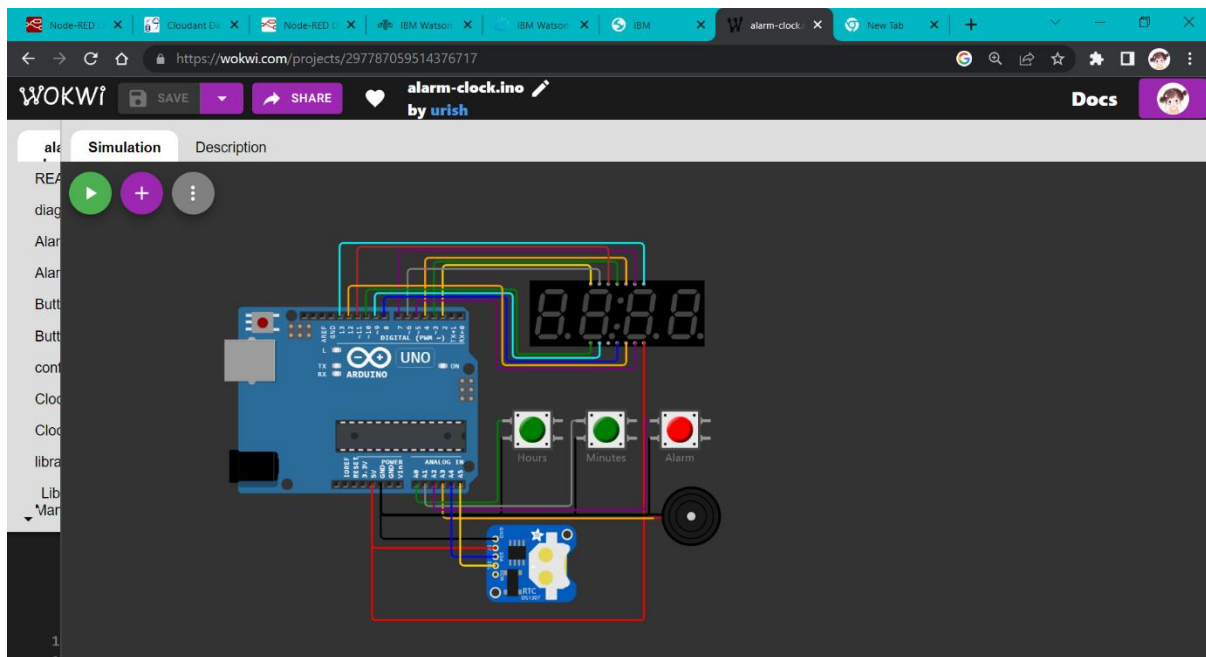
EQUIPMENTS REQUIRED:

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,
};
```

```cpp
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);
```

```cpp
  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;
```
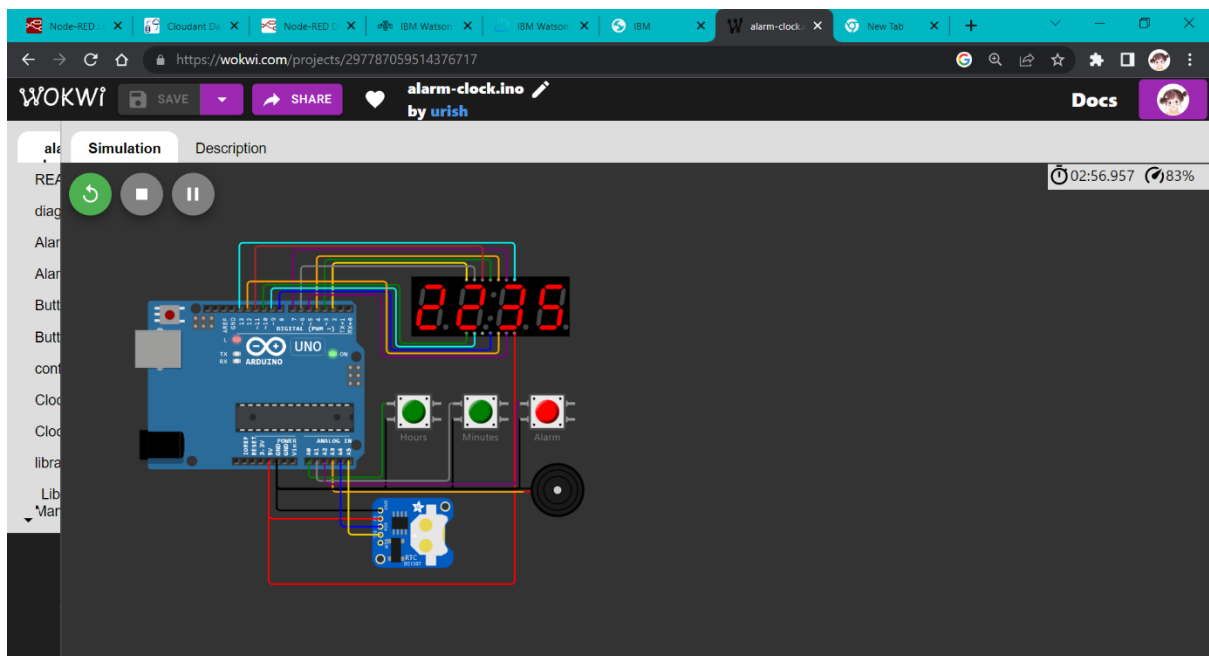
```
    case DisplaySnooze:

      snoozeState();

      break;

  }

}
```
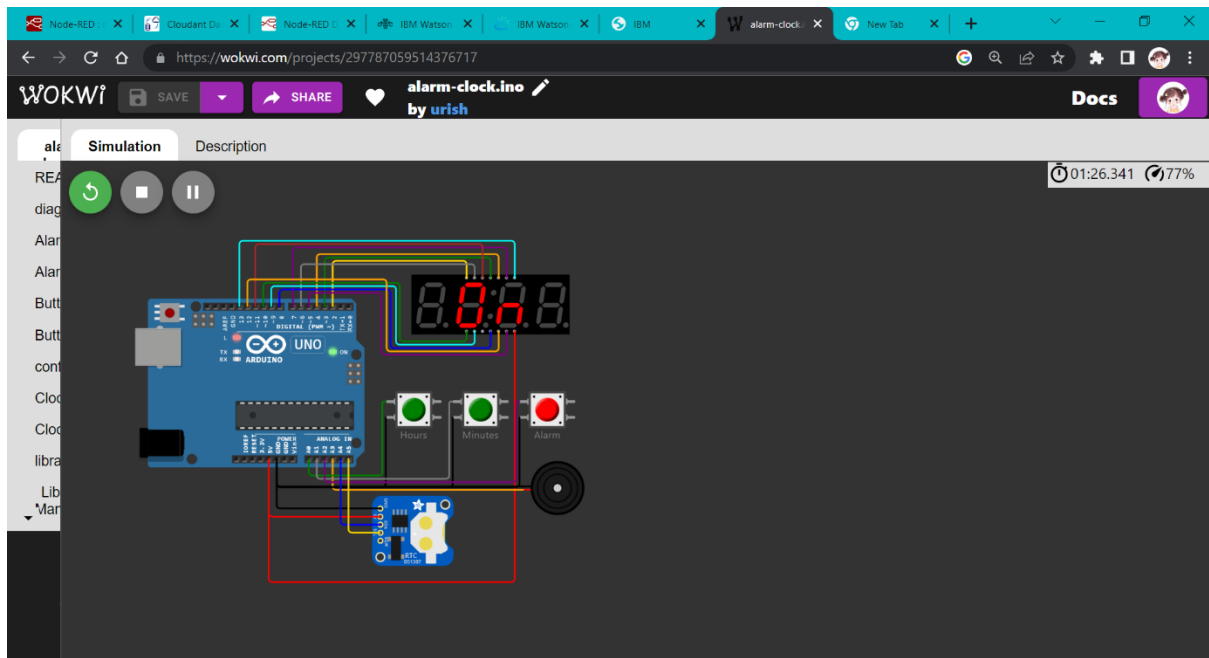
OUTPUT:

❖ At first it shows the current time.

❖ "ON" the alarm for setting specific time.

- ❖ To set the time, press the Minute/Hour buttons.Pressing the Alarm button enables/disables the alarm. The screen will display the alarm state by showing the word "on" or "off".

- ❖ After enabling the alarm, the current alarm time will be displayed for a few seconds. You can use the Minute/Hour buttons to adjust the alarm time.

- ❖ To finish, press the Alarm button again, or just wait a few seconds.

- ❖ Once we set the alarm it starts ringing.
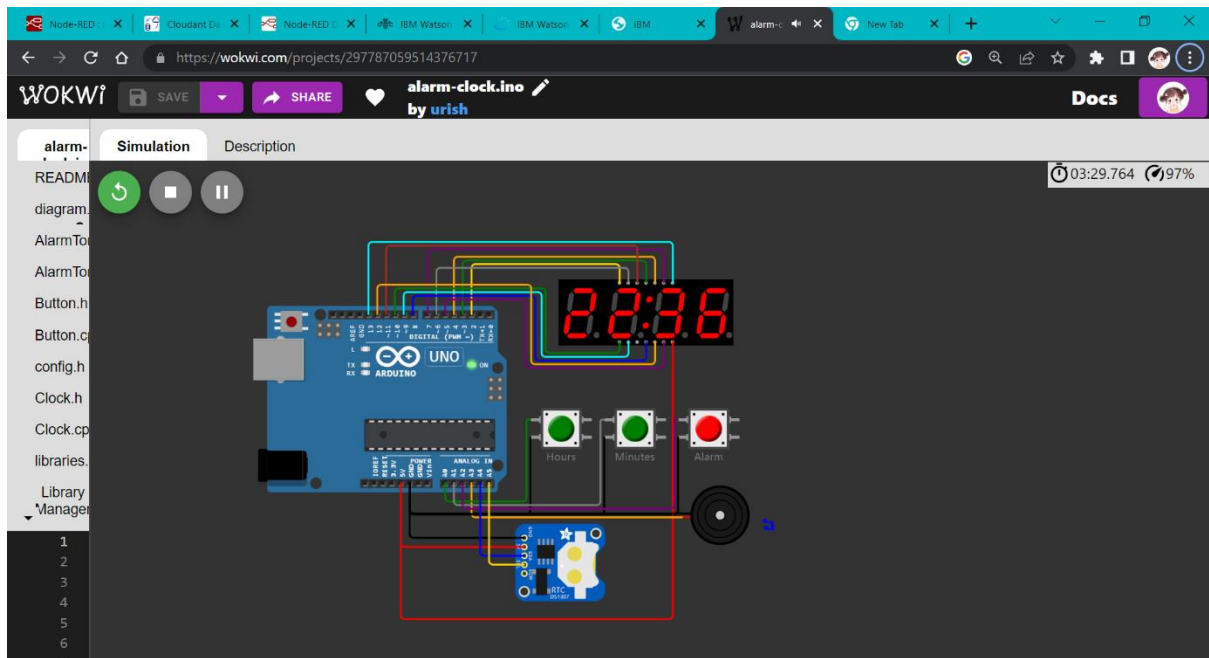
RESULT:

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