ALZHEIMER'S VIRTUAL ASSISTANT

Project Description:

Alzheimer's disease is a progressive disorder that causes brain cells to degenerate and die. It is the most common form of dementia.

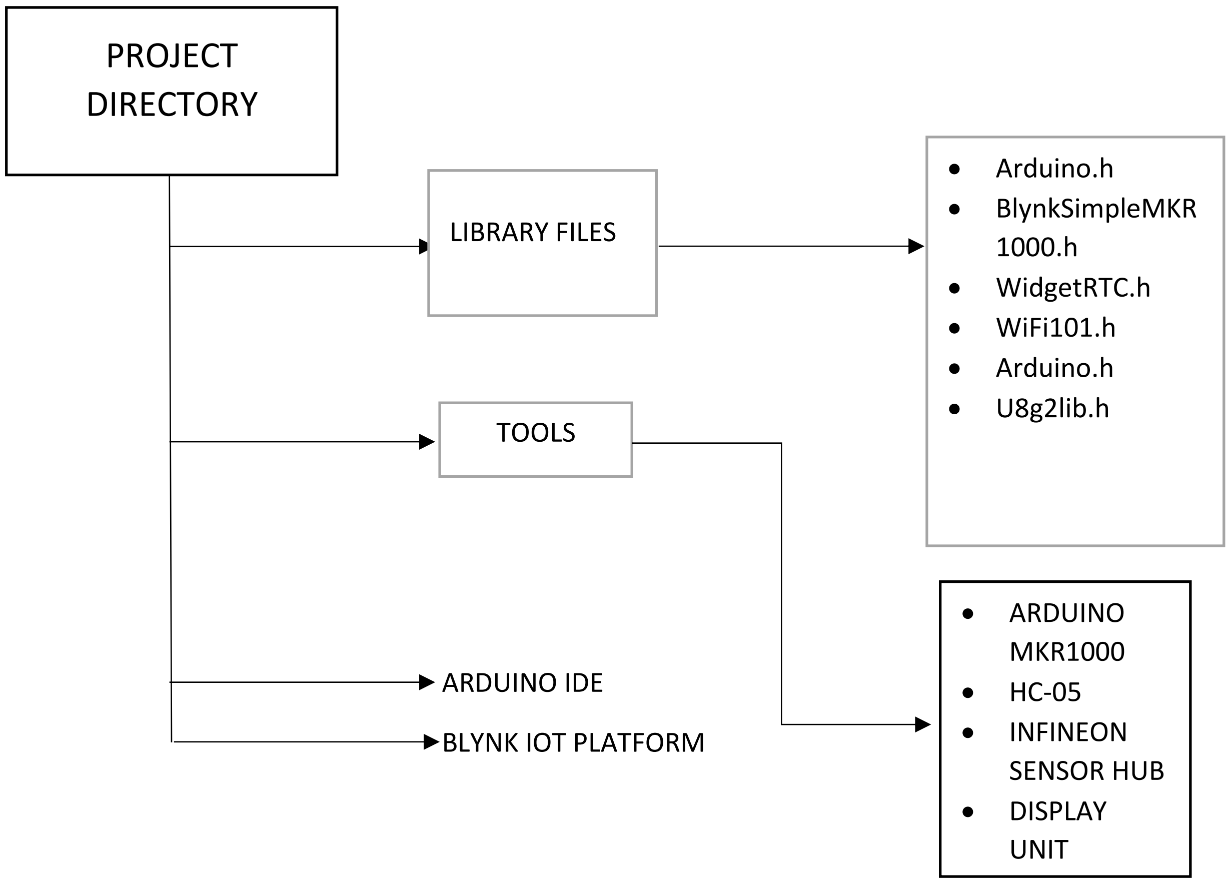
Worldwide, around 50 million people have dementia, and there are nearly 10 million new cases every year. In India, more than 1 million cases are reported per year.

This disease is mostly found in persons aged 65 or above. The patients affected by this disease should be treated with utmost care but lack of attention may lead to unexpected deaths.

To tackle these situations and to help the patients and caretakers, we have come with an idea of ALZHEIMER'S VIRTUAL ASSISTANT.

This is a wearable device which reminds the patient about the tasks like medications, exercises and other activities. It also alerts caretakers in Emergency situations and will also monitor the patients. In addition to this, we have incorporated health monitoring parameters so that it would be a complete medical aid for the patients

Project Directory



AT_Commands for HC-05

```
// Connect the HC-05 module and communicate using the serial monitor
//
// The HC-05 defaults to commincation mode when first powered on.
// Needs to be placed in to AT mode
// After a factory reset the default baud rate for communication mode is 38400

char c = ' ';

void setup() {
  // start the serial communication with the host computer
  Serial.begin(9600);
  Serial.println("Arduino with HC-05 is ready");

  // start communication with the HC-05 using 38400
  Serial1.begin(38400);
  Serial.println("Serial1 started at 38400");
}

void loop() {
  // Keep reading from HC-05 and send to Arduino Serial Monitor
  if (Serial1.available())
  {
    c = Serial1.read();
    Serial.write(c);
  }

  // Keep reading from Arduino Serial Monitor and send to HC-05
  if (Serial.available())
```

```
 {
   c =  Serial.read();

   // mirror the commands back to the serial monitor
   // makes it easy to follow the commands
   Serial.write(c);
   Serial1.write(c);
 }
}
```

Configuring Arduino MKR1000

```
//Including required libraries
#include <BlynkSimpleMKR1000.h>
#include <WidgetRTC.h>
#include <WiFi101.h>
#include <Arduino.h>
#include <U8g2lib.h>
#include <SPI.h>
#include<TimeLib.h>

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = ""; //Enter your Blynk auth token here

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "";
char pass[] = "";

//Defining Sensor Hub Nano board commands
#define HELLO "$hello id="
#define INFO  "$info id="
#define SENSOR_INFO "$sinfo id=1"
#define LOW_ENERGY      "$set_mode sid=1;md=mode;val=bg"
#define STANDARD_MODE   "$set_mode sid=1;md=prs_osr;val=16"
#define HIGH_PRECISION  "$set_mode sid=1;md=prs_mr;val=32"
#define START "$start_sensor id=1"
#define STOP  "$stop id="

//Defining fall and clearance thresholds
//You may need to change them, but I found these values to be good
#define FALL 0.7
#define CLEARANCE 0.2

//Defining Blynk virtual pins
#define vTEMPERATURE_PIN V0
#define vPRESSURE_PIN V1
#define vALTITUDE_PIN V2
#define vEVENTOR_PIN V3
#define vFALL_PIN V4

//Declaring required variables
float t, p, a, previousA;

//Boolean which tells tells if a fall is detected or not
boolean fallState;

//Variables needed for the fall detection algorithm
```

```
unsigned long previousMillis = 0;

const long interval = 1000;


//BTconnected is false when not connected and true when connected

boolean BTconnected = false;


//Defining BT state and LCD backlight pins

int btStatePin = 9;

int backlightPin = 2;


BlynkTimer timer;

WidgetRTC rtc;


//Nokia 5110 Display wiring

U8G2_PCD8544_84X48_F_4W_SW_SPI u8g2(U8G2_R0, /* clock=*/ 7, /* data=*/ 8, /* cs=*/ 3, /*
dc=*/ 5, /* reset=*/ 4);


void setup() {

  //Initialize both serial ports:

  Serial.begin(115200);

  Serial1.begin(115200);


  //Setup the timed fuctions

  timer.setInterval(1000L, sendSensorValues);

  timer.setInterval(3000L, showTimeAndDate);


  //Setting up required inputs and outputs

  pinMode(btStatePin, INPUT);

  pinMode(backlightPin, OUTPUT);

  digitalWrite(backlightPin, LOW);
  u8g2.begin();

  showStartMessage();
  delay(2000);

  // wait until the bluetooth module has made a connection
  while (!BTconnected) {
    if (digitalRead(btStatePin) == HIGH)  {
      BTconnected = true;
    }
    else {
      showWaitingFor();
    }
  }

  initSensorHub();

  Blynk.begin(auth, ssid, pass);
  rtc.begin();

  setBlynkWidgets();
  showTimeAndDate();
  sendCommand(START);
}

void loop() {
  Blynk.run();
  timer.run();
  getSensorValues();
  checkIfFalling();
}
```

```
void sendCommand (String sensorCommand) {
  //This function sends commands through the bluetooth module on the hardware serial port to the
the Sensor Hub Nano
  //For example: "sendCommand(START);", starts the flow of data from the sensor
  //The full list of commands I know are defined at the top of the sketch
  Serial1.println(sensorCommand);
}

void initSensorHub() {
  //Initialise the Sensor Hub Nano, and give an error if there is any problem
  String junkVal;
  sendCommand(INFO);
  while (Serial1.find("IFX_NanoHub") == false) {
    sendCommand(INFO);
    Serial.println("ERROR");
    showErrorMessage();
  }
  junkVal = Serial1.readStringUntil('\n');
  junkVal = "";
  showConnectedMessage();
  delay(1500);
}

void getSensorValues() {
  //Retrieves the sensor values from the Sensor Hub Nano through the Serial1 port
  String junkVal;
  if (Serial1.available()) {
    junkVal = Serial1.readStringUntil('\n');

    junkVal = Serial1.readStringUntil('t');
    t = Serial1.parseFloat();


    junkVal = Serial1.readStringUntil('p');
    p = Serial1.parseFloat();


    junkVal = Serial1.readStringUntil('a');
    a = Serial1.parseFloat();


    junkVal = Serial1.readStringUntil('\n');
  }
}


void sendSensorValues() {
  //Sending the sensor values to the Blynk server
  Blynk.virtualWrite(vTEMPERATURE_PIN, t);
  Blynk.virtualWrite(vPRESSURE_PIN, p);
  Blynk.virtualWrite(vALTITUDE_PIN, a);
}


void checkIfFalling() {
  //Algorithm to check if the patient is falling
  unsigned long currentMillis = millis();
  if ((currentMillis - previousMillis) >= interval) {
    float diff = previousA - a;
    if ((diff >= (FALL - CLEARANCE)) && (diff <= (FALL + CLEARANCE))) {
      fallState = true;
      //Here insert what you need to do if fall is detected, such as sending a notification or email with
Blynk
      //Or you could also use IFTTT to call or send an sms to alert the caretaker (more info in the
project documentation)
      Serial.println("Falling");
      showFallMessage();
      //In this example, vFALL_PIN (virtual pin 4) is set to 255 if fall is detected
      Blynk.virtualWrite(vFALL_PIN, 255);
      //You can send a notification using only the notification widget too!
      //Blynk.notify("DPS310 detected a fall!");
    }
    previousA = a;
    previousMillis = currentMillis;
    fallState = false;
    //Set vFALL_PIN to 0 if a fall isn't detected
    Blynk.virtualWrite(vFALL_PIN, 0);
  }
}

void showStartMessage() {
  //Shows the start-up message
  u8g2.clearBuffer();
  u8g2.drawRFrame(3, 7, 75, 31, 7);
  u8g2.setFont(u8g2_font_prospero_bold_nbp_tf);
  u8g2.drawStr(8, 19, "Alzheimer's");
  u8g2.drawStr(12, 35, "Assistant");
  u8g2.sendBuffer();
}

void showWaitingFor() {
  //Shows the waiting for Sensor Hub Nano message
  u8g2.clearBuffer();
  u8g2.setFont(u8g2_font_prospero_bold_nbp_tf);
  u8g2.drawStr(9, 15, "Waiting for");
  u8g2.drawStr(8, 28, "Sensor Hub");
  u8g2.drawStr(22, 41, "Nano !!!");
  u8g2.sendBuffer();
```

```
}

void showConnectedMessage() {
  //Shows the connected message
  u8g2.clearBuffer();
  u8g2.setFont(u8g2_font_7x13B_tf);
  u8g2.drawStr(0, 10, "Connected to");
  u8g2.drawStr(8, 22, "Infineon's");
  u8g2.drawStr(7, 34, "Sensor Hub");
  u8g2.drawStr(29, 46, "Nano");
  u8g2.sendBuffer();
}

void showErrorMessage() {
  //Shows the error message
  u8g2.clearBuffer();        // clear the internal memory
  u8g2.setFont(u8g2_font_fub14_tf); // choose a suitable font
  u8g2.drawStr(9, 30, "ERROR"); // write something to the internal memory
  u8g2.sendBuffer();         // transfer internal memory to the display
}

void showSensorValues() {
  //Shows the sensor values on the display
  char bufT[10];
  char bufP[10];
  char bufA[10];

  String(t).toCharArray(bufT, 10);
  String(p).toCharArray(bufP, 10);
  String(a).toCharArray(bufA, 10);
  u8g2.clearBuffer();
  u8g2.setFont(u8g2_font_6x10_tf );
  //Display the temperature
  u8g2.drawStr(0, 10, "T:");
  u8g2.drawStr(12, 10, bufT);
  u8g2.drawStr(73, 10, "C");
  u8g2.drawCircle(70, 4, 1, U8G2_DRAW_ALL);
  u8g2.drawHLine(0, 12, 85);
  //Display the pressure
  u8g2.drawStr(0, 26, "P:");
  u8g2.drawStr(12, 26, bufP);
  u8g2.drawStr(60, 26, "mBar");
  u8g2.drawHLine(0, 28, 85);
  //Display the altitude
  u8g2.drawStr(0, 42, "A:");
  u8g2.drawStr(12, 42, bufA);
  u8g2.drawStr(72, 42, "m");
  u8g2.drawHLine(0, 44, 85);
  //Send the values to the display
  u8g2.sendBuffer();
}

void showFallMessage() {
  //Show the fall detected message
  u8g2.clearBuffer();
  u8g2.setFont(u8g2_font_7x13B_tf);
  u8g2.drawStr(27, 20, "Fall");
  u8g2.drawStr(13, 32, "Detected!");
  u8g2.sendBuffer();
  delay(1000);
}
```

```
void showPillReminder() {
  //Show the pill reminder message
  u8g2.clearBuffer();
  u8g2.setFont(u8g2_font_7x13B_tf);
  u8g2.drawStr(0, 20, "Time to take");
  u8g2.drawStr(5, 32, "your pills!");
  u8g2.sendBuffer();
}

void showExerciseReminder() {
  //Show the exercise reminder message
  u8g2.clearBuffer();
  u8g2.setFont(u8g2_font_7x13B_tf);
  u8g2.drawStr(16, 20, "Time to");
  u8g2.drawStr(12, 32, "exercise!");
  u8g2.sendBuffer();
}

void showTimeAndDate() {
  //Displays the time and date from the RTC widget in Blynk in 24 hours format
  if (year() == 1970) {
    //Serial.println("Time not yet synced");
  }
  else if (year() != 1970) {
    char bufHours[3];
    char bufColon[2];
    char bufMinutes[3];
    char bufDate[11];

    String currentHours = String(hour());
    String colon = ":";
    String currentMinutes = String(minute());
    String currentDate = String(day()) + "/" + month() + "/" + year();

    String(currentHours).toCharArray(bufHours, 3);
    String(colon).toCharArray(bufColon, 2);
    String(currentMinutes).toCharArray(bufMinutes, 3);
    String(currentDate).toCharArray(bufDate, 11);

    u8g2.clearBuffer();
    u8g2.setFont(u8g2_font_inr33_mf);
    u8g2.drawStr(30, 30, bufColon);
    u8g2.setFont(u8g2_font_logisoso32_tn);
    u8g2.drawStr(0, 32, bufHours);
    u8g2.drawStr(45, 32, bufMinutes);
    u8g2.setFont(u8g2_font_saikyosansbold8_8n);
    u8g2.drawHLine(0, 35, 85);
    u8g2.drawStr(0, 46, bufDate);
    u8g2.sendBuffer();
  }
}

BLYNK_WRITE(vEVENTOR_PIN) {
  //Use the Eventor widget to check if it's time to do a task (take medication in this case)
  int currentValue =  param.asInt(); // 0 to 1
  if (currentValue == 1) {
    showPillReminder();
    //Serial.println("Time to take your pills");
  }
  else {
    //Serial.println("Not the time to take pills");
  }
}

void setBlynkWidgets() {
  //This sets the colour of each widget in the Blynk app
  //You may remove this from the sketch if you want to set colours manually through the Blynk app
  //You could also specifiy the hex value of each colour you need

  //Set temperature widget color to white
  Blynk.setProperty(vTEMPERATURE_PIN, "color", "#FFFFFF");

  //Set pressure widget color to blue
  Blynk.setProperty(vPRESSURE_PIN, "color", "#00BBFF");

  //Set altitude widget color to yellow
  Blynk.setProperty(vALTITUDE_PIN, "color", "#FFFF00");
}

BLYNK_CONNECTED() {
  //Synchronize time on connection, if connection drops
  rtc.begin();
}
```