Russel Spiro
Robotics for Disabilities Term Project

A Remote-Controlled Thermostat for the Elderly and Visually Impaired

Introduction:

The spreading of technology into mainstream markets has much potential to effect the industry of elderly care. Internet of Things allows smart sensors, wearable devices, and house appliances to all connect to a central system, creating for more efficient communication with caregivers as well as an increased sense of independence for the elderly living on their own. Although this has great potential to transform the world of elderly care, the elderly population has shown significant resistance to allowing new technology into their homes. To a person unfamiliar with new trends, the financial cost and difficulty of installing such devices outweighs the benefits[1][2]. In addition, they are often concerned with their privacy when considering sensors and monitoring devices that broadcast information to caretakers. Therefore there are currently several steps that still can be taken to better address the demands of the elderly population.

Several simpler devices have been developed to improve more specific aspects of daily life while minimizing the complication and cost of use. One such device is a talking thermostat that allows the visually impaired to change and monitor the temperature of their house without the need to look at a dial. Devices like the VIP3000 will report the settings back to the user with a simple touch of a button[3].

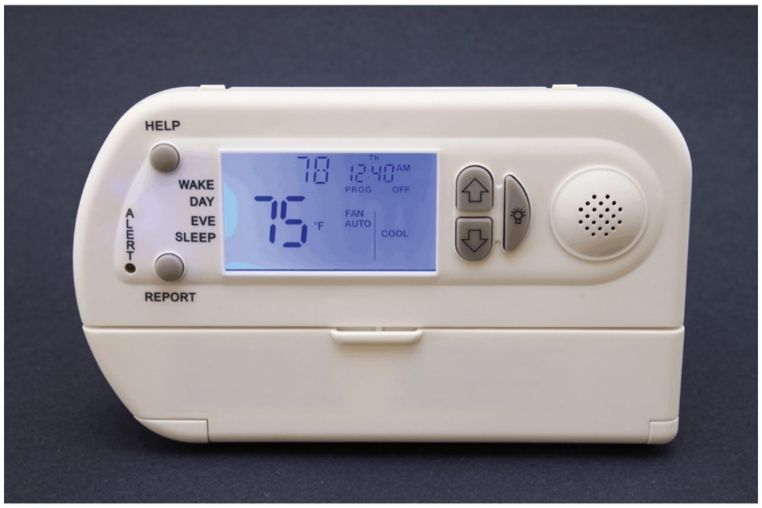


Figure 1 The VIP3000

Pressing the Report button plays back the indoor temperature, the temperature settings, and the current date and time. Any change in setting is also reported whenever the up and down buttons are pressed.

This project is a prototype for a device that attempts to bring some of the benefits of IoT into a simpler and less intimidating package. An app was developed that connects over Bluetooth to a controller equipped with a fan and temperature sensor. Using a multicolored and easily visible interface, the user can adjust temperature settings as well as have the settings read back through the phone.

Materials:

MIT App Inventor was used to develop the app. The Bluetooth Low-Energy block library was used in addition to the standard block libraries

The air conditioning unit consists of an RFduino, a coin battery shield, and a 3V motor powered by 2 AA batteries and an Adafruit motor driver.

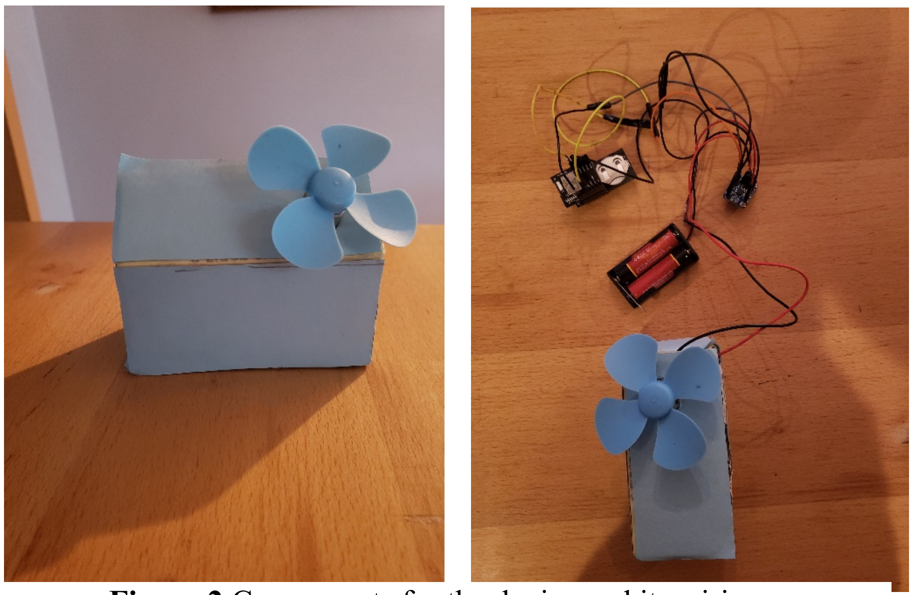


Figure 2 Components for the device and its wiring.

The power supplies, RFduino, and motor driver, shown on the right, all fit inside the box with the 3V fan.

<u>Code Overview:</u>

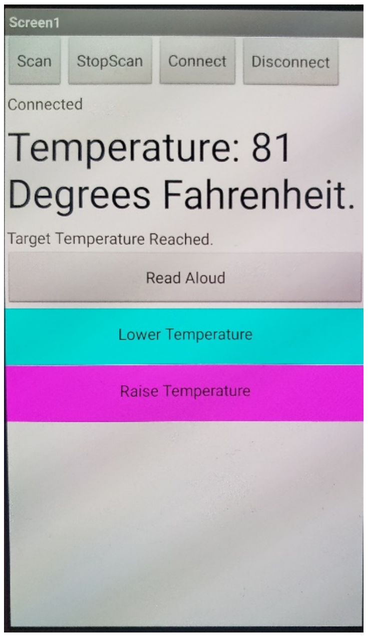
MIT App Inventor:

First, the app connects to the RFduino via BLE. A subscription is made to receive floats.

Temperature is received from the RFduino and displayed on the screen. The target temperature is initialized to the room temperature. Whenever the target temperature is not equal to the room temperature, it is displayed on the screen

Wait for one of 3 possible button-presses: “Raise Temperature” , “Lower Temperature”, and “Read Aloud”. See figure 2 below.

See the appendix for the RFduino code.



<u>**Figure 3**: User Interface for the app to control the air conditioning device.</u>

At the top are controls to connect and disconnect from Bluetooth. The "Read Aloud" button causes the phone to report the current room temperature, the target temperature, and whether the fan is on or off. The “Lower Temperature” and “Raise Temperature” buttons allow the user to lower and raise the target temperature, respectively. Whenever either of these buttons are pressed, a corresponding announcement is made to the user.

Conclusions:

This design succeeded as a prototype for a mobile-controlled talking thermostat. It contains the basic functionality of a standard talking thermostat like the VIP3000 but with the benefit of being controlled on the phone. The user can control the desired temperature, and the fan will turn on if the room temperature is too hot and turn off if the room temperature is too cold. Any time the fan switches between on and off, an announcement is spoken through the phone. In addition, the status of the device, including the current room temperature, the current target temperature, and the state of the fan can be read at any time by pressing the "Read Aloud" button. Using a smartphone to control the thermostat can be a significant improvement over other talking thermostats which require the user to be in front of the device itself. With other stand-alone devices, the individual must push the buttons on the device itself to receive any audio feedback. With a smartphone, the temperature can be both set and reported at ease from a distance. This could help an aging individual for whom moving around the house can be difficult or painful. Advanced IoT systems like Alexa may be a more robust option for having interconnectivity throughout the house, but a simpler single-function device such as this should have a lower cost and perceived difficulty of use for the elderly. Another strength of this device is that the user interface is more colorful and visually stimulating than that of a physical thermostat, potentially making it a better option for visually impaired people who can see a screen in front of them but have a hard time locating small buttons.

There are several improvements that can make this device more suitable for the market. Adding voice-recognition capability would make it significantly more accessible for the blind and visually impaired. In addition, the small fan should be replaced by a larger cooling device that could make a change in the room temperature on its own. The small 3V fan currently does not produce more than a breeze and often does not produce enough torque to spin on its own. Finally, a heating element should be added and tested to ensure that full temperature control can be achieved.

References:

[1] Pal, D., Funilkul, S., Charoenkitkarn, N., & Kanthamanon, P. (2018). Internet-of-Things and Smart Homes for Elderly Healthcare: An End User Perspective. *IEEE Access,6*, 10483-10496.

[2] Paul, F. (2018, August 02). Why IoT for seniors is a lot tougher than it looks. Retrieved from https://www.networkworld.com/article/3294198/internet-of-things/why-iot-for-seniors-is-a-lot-tougher-than-it-looks.html

[3] Dave. (2018, December 10). Best Thermostats For The Visually Impaired -Buyers Guide & Reviews (2018). Retrieved from https://mythermostatreviews.com/best-thermostats-visually-impaired/

Appendix:

RFduinoTempRead.ino

```
#include <RFduinoBLE.h>


#define OUT1 5
#define OUT2 6
float temp;
float targetTemp;
float prevTemp;
char test[14];
boolean first;
// This function is called only once, at reset.
void setup()
{
        // Enable serial debug.
        Serial.begin(9600);
        Serial.println("RFduino example started");
        Serial.println("Serial rate set to 9600 baud");

        // Enable outputs.
  pinMode(OUT1, OUTPUT);
  pinMode(OUT2, OUTPUT);




        // Check RFduino CPU temperature, and print to log
        float CPU_temperature = RFduino_temperature(CELSIUS);
        Serial.print("RFduino_temperature is: ");
        Serial.print(CPU_temperature);
        Serial.println(" deg C");

        // this is the data we want to appear in the advertisement
        // (the deviceName length plus the advertisement length must be <= 18 bytes
        // RFduinoBLE.advertisementData = "ledbtn";
        RFduinoBLE.advertisementInterval = 500;
        Serial.println("RFduino BLE Advertising interval 500ms");
        RFduinoBLE.deviceName = "RFduino";
        Serial.println("RFduino BLE DeviceName: RFduino");
        RFduinoBLE.txPowerLevel = -20;
        Serial.println("RFduino BLE Tx Power Level: -20dBm");

        // start the BLE stack
        RFduinoBLE.begin();
        Serial.println("RFduino BLE stack started");
  temp = round(RFduino_temperature(FAHRENHEIT));
  targetTemp = temp;
  prevTemp = temp;
 first = true;
}

// This function is called continuously, after setup() completes.
void loop()
{
  temp = round(RFduino_temperature(FAHRENHEIT));
  if(temp != prevTemp){//Detect change in outside temperature
   delay(500);// Delay to keep value from skipping
   //Don't look at large jumps in temperature- this is noise from the sensor
   if(temp > prevTemp){
     if((temp - prevTemp) > 1){
       temp = prevTemp;
     }
   }
   if(temp < prevTemp){
     if((prevTemp - temp) > 1){
       temp = prevTemp;
     }
   }
   //After adjusting for noise, check to see if fan should but turned on or off
   checkFan();
  }
  prevTemp = temp;


         RFduino_ULPDelay( SECONDS(0.5) );


 // get a cpu temperature sample
 // degrees c (-198.00 to +260.00)
 // degrees f (-128.00 to +127.00)


 RFduinoBLE.sendFloat(temp);
 Serial.println(targetTemp);
```

```
}



void RFduinoBLE_onAdvertisement()
{
        Serial.println("RFduino is doing BLE advertising ...");
        digitalWrite(RED_LED_PIN, LOW);
        digitalWrite(GREEN_LED_PIN, LOW);
        digitalWrite(BLUE_LED_PIN, LOW);
}

void RFduinoBLE_onConnect()
{
  targetTemp = temp;
  prevTemp = temp;
        Serial.println("RFduino BLE connection successful");
        digitalWrite(RED_LED_PIN, LOW);
        digitalWrite(GREEN_LED_PIN, HIGH);
        digitalWrite(BLUE_LED_PIN, LOW);
}

void RFduinoBLE_onDisconnect()
{
        Serial.println("RFduino BLE disconnected");
        // don't leave the leds on after disconnection
        digitalWrite(RED_LED_PIN, LOW);
        digitalWrite(GREEN_LED_PIN, LOW);
```

```
            digitalWrite(BLUE_LED_PIN, LOW);
}


void RFduinoBLE_onReceive(char *data, int len)
{
  first = false;



            // if the first byte is 0x01 / on / true
            Serial.println("Received data over BLE");
// Serial.println(data);

            if (data[2] && (data[1] == 0))   //normally data[0]  Send[0, 1]
            {
  //fanOff();
                        Serial.println("Increasing Temp");
   data[1] = 0;
   targetTemp++;
//temp = RFduino_temperature(CELSIUS);
            }
 else if(data[0]){
  Serial.println("Reading");
 // temp = RFduino_temperature(CELSIUS);
  data[0] = 0;
 }
            else
            {
                        Serial.println("Cooling Off");
```

```
    targetTemp--;
    //fanOn();
            }


 checkTemp();
}




void fanOn(){
  Serial.println("fan on");
    digitalWrite(OUT1, HIGH);
  digitalWrite(OUT2, LOW);
}

void checkTemp(){
  if (targetTemp < (temp)){
  fanOn();
  }
  else if (targetTemp >= (temp)){
    fanOff();
  }
}

void checkFan(){// For if outside temperature is changing
  if (targetTemp < (temp)){
  fanOn();
  }
  else if (targetTemp >= (temp)){
    fanOff();
  }
}



void fanOff(){
  Serial.println("fan off");
    digitalWrite(OUT1, LOW);
  digitalWrite(OUT2, LOW);
}
```