ELEKTRONICA-ICT

Elektronische systemen 2 2023-2024

OneWire

Authors *Mauro De Bruyn*
Thibe Van Orshaegen

Abstract

This project will focus on the development of an access control system integrated with person detection using an infrared motion sensor. Using the OneWire communication protocol, the system reads I-Button user inputs.

The primary hardware components include an Arduino Uno R3 as the master device and an Arduino Nano 33 BLE Sense as the slave device. The Arduino Uno R3 manages the I-Button inputs, motion sensor (SE-10) and a 1.8 TFT screen for displaying data and sensor values. A DS3231 real-time clock (RTC) module is used to maintain accurate timekeeping, even during power outages, and display date and time when an I-Button is scanned. the Arduino Nano 33 BLE Sense provides sensor data through I2C communication, including temperature, humidity, and XYZ coordinates.

The system's design aims to enhance security applications such as security safety checks in specific areas.

The project’s details such as schematics, coding details and case design are documented in this application note to get an understanding of the setup and its functionalities.

Table of Contents

1 Introduction

The goal of this project is to create a security control system that integrates motion detection and monitoring capabilities. The system uses OneWire communication to read I-Button sensors. The main microcontroller uses the OneWire library to decode the signal into a code.

An Arduino Uno R3 is used as the main controller and interfaces with most hardware including a motion sensor (SE-10) for detecting movement and a 1.8 TFT screen for displaying sensor readings and I-Button information.

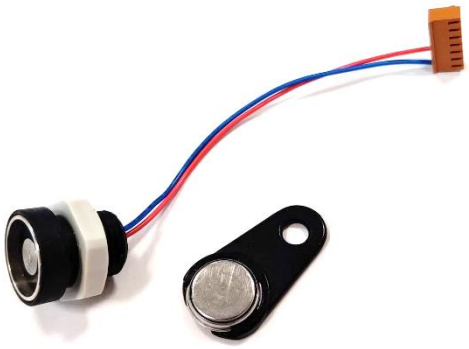
To provide additional data an Arduino nano 33 BLE is incorporated as a slave device, communicating sensor data (temperature, humidity and XYZ coordinates) to the master device using I2C protocol.

Additionally, a real time clock ensures accurate timekeeping which is essential for timestamping I-Button detection.

2 Material and Methods

2.1 I-Button

The I-button sensor is part of the OneWire data communication system. When a user holds the button to the sensor, the voltage is pulled down in a specific sequence. In the code, a function in the OneWire library decodes this signal and reads the unique code of the button. Our project includes three sensors connected to an Arduino. This system is particularly useful for example security firms to verify that their personnel have checked all areas within a building.



2.2 Arduino

2.2.1 UNO R3

This will be the main board which hosts the main code. It scans the I-button sensors, reads the motion sensor and outputs them to a display.

We chose the UNO R3 because the OneWire library did not seem to function on the Arduino nano 33 BLE sense. The requirements for this project were to also read sensor values but this board does not have any sensors. Therefore, we connected tis board to the Arduino nano 33 BLE to read its sensor values over I2C communication.

2.2.2 Nano 33 BLE Sense

The Arduino Nano 33 BLE Sense was used as a slave device due to the absence of built-in sensors in the Arduino UNO R3. We developed code to collect sensor data from the embedded sensors on the Nano 33 BLE sense and send this data over an I2C connection to the master device, the Arduino UNO R3. The collected sensor data includes temperature, humidity, and the XYZ coordinates of the board.

2.3 SE-10

The SE-10 motion sensor is a passive infrared (PIR) detector designed for detecting motion from infrared light sources, typically at wavelengths emitted by humans. This sensor is commonly used in security systems, such as alarm systems and automatic lighting controls.

Two pyroelectric sensing elements allow the sensor to see moving infrared light sources while ignoring changes in the background. When the sensor is powered, it must calibrate itself to the background infrared radiation before it can detect motion, it usually takes about 1-2 seconds.

2.4 DS3231

The DS3231 real-time clock (RTC) is designed for use with Raspberry Pi and Arduino systems. It features automatic voltage adjustment, enabling operation of 3.3V and 5V environments. One of the standout features of the DS3231 is its battery backup, which ensures continuous timekeeping even when the main power supply is interrupted. This functionality is crucial for maintaining accurate time data during power outages.

2.5 1.8 TFT Screen

This small and compact screen is designed for Arduino and Raspberry pi. It has a 128 x 160-pixel resolution and holds an ST7735S driver chip.

The screen works with an SPI interface. The serial peripheral interface (SPI) is an interface bus that is used to send data between microcontrollers, in this case it will send data from the Arduino uno R3 to the TFT screen.

The controller is operated with 5V and the backlight is supplied by 3.3V.

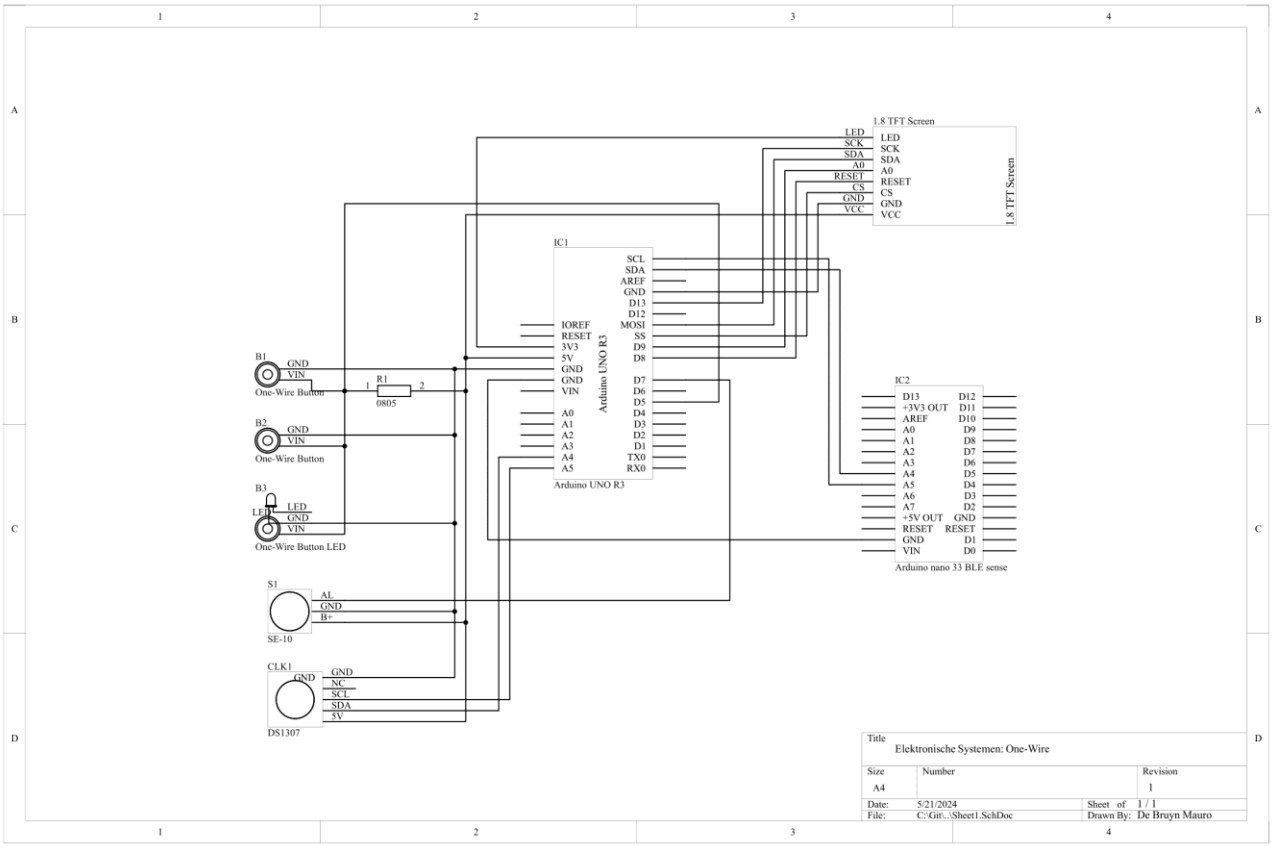
This specific screen can hold an SD card to store data such as background images, so the screen does not need data from a pc.

3 Results

Beperk het schrijven tot effectief uitgevoerd werk en zonder opinie, want deze komt onder hoofdstuk 4.

3.1 Schematics

The photo shows the schematic design for this project, it shows the full project with 2 Arduino's.



3.2 Code master device

The following code is uploaded onto the Arduino Uno R3, this is the main controller that is connected to the motion sensor, OneWire buttons and the 1.8 TFT screen.

The code explained below can be found on the GitHub repository.

3.2.1 timeStamp

This function will be called when the user scans an I-Button. It pulls the date and time from the RTC real time clock that is connected to the Arduino uno R3 and prints it on the TFT screen.

```
29  void timeStamp(void){
30    DateTime now = rtc.now();
31    // Write date and time information
32    TFTscreen.text("Date & Time: \n", 0, 0);
33    TFTscreen.text(String(now.day(), DEC).c_str(), 0, 10);
34    TFTscreen.text("/", 23, 10);
35    TFTscreen.text(String(now.month(), DEC).c_str(), 37, 10);
36    TFTscreen.text("/", 50, 10);
37    TFTscreen.text(String(now.year(), DEC).c_str(), 60, 10);
38    TFTscreen.text(daysOfTheWeek[now.dayOfTheWeek()], 0, 20);
39    TFTscreen.text(String(now.hour(), DEC).c_str(), 0, 30);
40    TFTscreen.text(":", 23, 30);
41    TFTscreen.text(String(now.minute(), DEC).c_str(), 35, 30);
42    TFTscreen.text(":", 57, 30);
43    TFTscreen.text(String(now.second(), DEC).c_str(), 67, 30);
44
45    delay(1000); // delay 1 second
46  }
```

3.2.2 receiveEvent

This code will enable the master device to receive sensor data from the slave device. It will parse the list of data using a case statement and store it in the corresponding variable. When the variable is stored, it sets a bool to “true”.

```
62  // Function to handle received data
63  void receiveEvent(int bytes,bool een,bool twee,bool drie,bool vier,bool vijf) {
64    while (Wire.available()) { // While data is available to receive
65      Wire.readBytes((uint8_t*)&sensorData, sizeof(sensorData)); // Read the incoming sensor data
66
67      getal += 1;
68
69      //Serial.println(sensorData);
70      //Serial.println(getal);
71      Serial.println(getal);
72
73      switch(getal) {
74        case 1:
75          temp = sensorData;
76          Serial.print("Temp: ");
77          Serial.println(temp);
78
79          //TFTscreen.setCursor(0, 75);
80          //TFTscreen.print("Temp: 27");
81          temperatuurbool = true;
82          //print(temperatuurbool);
83
84
85          break;
```

3.2.3 updateSensorValues

This function will update the screen when new sensor values are received.

It checks the status of the bool from the receiveEvent function. When the statement is “true” it means that the variable is filled with new data. The udateSensorValues function will than print the senor value to the TFT screen and set the boolean value to “false” to prevent it from printing previous outdated sensor values.

The photo shows the process of printing the updated value of the temperature sensor. This function also contains code for the humidity, x, y and z sensors.

```
132 void updateSensorValues(void){
133   TFTscreen.noStroke();
134   receiveEvent(10,temperatuurbool,humiditybool,xbool,ybool,zbool);
135     //print(temperatuurbool)
136     if(temperatuurbool == true)
137     {
138       //Clear section
139       TFTscreen.fill(0,0,0);
140       TFTscreen.rect(30, 71, 35, 15);
141 
142       TFTscreen.setCursor(0, 74);
143 
144       TFTscreen.print("Temp: ");
145       TFTscreen.print(temp);
146       //temperatuurbool.clear();
147       temperatuurbool = false;
148     }
```

3.2.4 logIButton

This function will run almost constantly and checks whether a sensor reads an I-button or not. When an I-button is detected, this function will decode the code using a function of the OneWire library and print it on the screen. It also recalls the timeStamp function and prints the output on the screen so users can check the date, time and I-Button code of the last check-in.

3.2.5 Motion

This loop will check if motion is detected. When it is detected, the loop will print “Motion detected!” on the TFT screen. When it is not detected the text will disappear.

```
301   int proximity = digitalRead(MOTION_PIN);
302   TFTscreen.line(0, 100, 200, 100);
303   TFTscreen.noStroke();
304     if (proximity == LOW) // If the sensor's output goes low, motion is detected
305   {
306     TFTscreen.stroke(20, 150, 255);
307     TFTscreen.line(0, 100, 200, 100);
308     Serial.println("Motion detected!");
309     TFTscreen.setCursor(0, 110);
310     TFTscreen.print("Motion detected!");
311   }
312   else
313   {
314     TFTscreen.line(0, 100, 200, 100);
315     TFTscreen.fill(0,0,0);
316     TFTscreen.rect(-1, 100, 240, 50);
317     //TFTscreen.rect(0, 110, 240, 10, TFTscreen.Color565(0, 0, 0)); // Clear the ar
318   }
319   TFTscreen.line(0, 100, 200, 100);
320   TFTscreen.stroke(255, 255, 255);
```

3.3 Code slave device

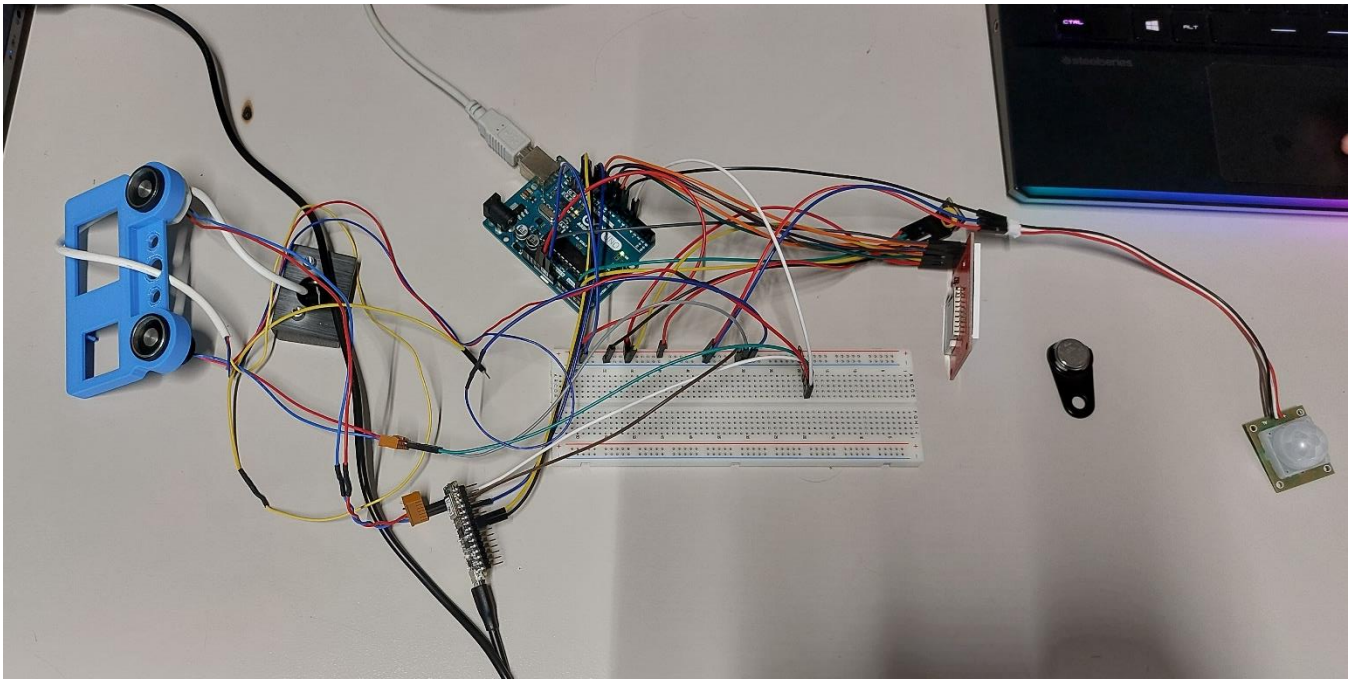
This code is hosted on the slave device which is the Arduino nano 33 BLE sense. This code communicates with the other microcontroller over the I2C protocol.

When a connection is discovered the device will send its current sensor values. After the first send it will wait 5 seconds. After the delay it will send a new list of values to the master device.

3.4 Setup

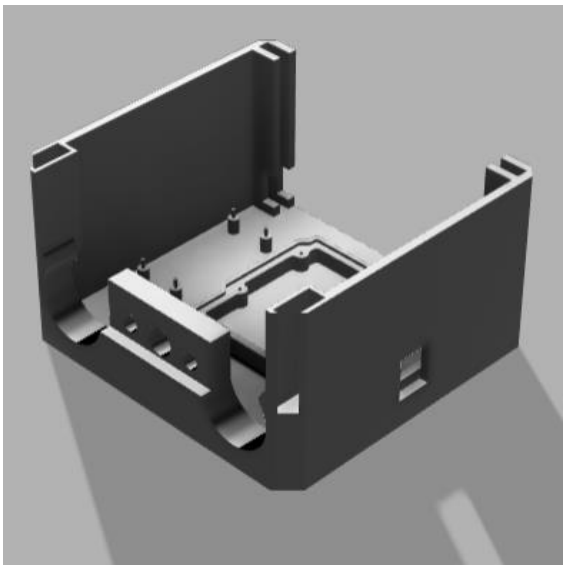
The photo shows the setup which we used to test code for this project, it shows the 3 I-button sensors, 1 I-Button, motion sensor and TFT screen connected to the Arduino UNO R3. The smaller Arduino nano 33 BLE sense is connected to the main board with I2C connection.

Both Arduino's are connected to a laptop to upload code and power the microcontrollers. It is possible in a further stage to power the Arduino's with an external power supply but for this project we just used laptops to power the boards.



3.5 Case design

3.5.1 Base



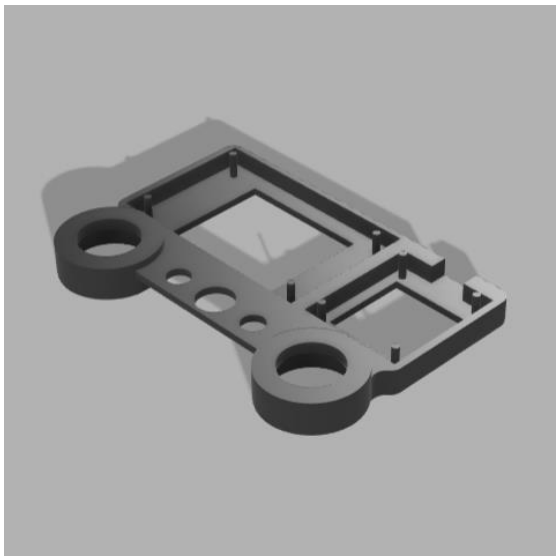
The picture on the right shows the base of the case. The 4 pins are reserved for the Arduino nano 33 BLE sense and fit into the holes of the Arduino. On the right side there is a cutout for the Arduino Uno, the board is large and heavy enough to hold itself into place. When using the project, the USB will be plugged in trough the hole in the right side of the case, this will also help to keep the board fixed into place.

3.5.2 Top

The top of the case will slide on the base and holds itself into place. The hole on the back is reserved for the USB cable that's needed to power the Arduino nano.

It also has a slot to host a small breadboard.

3.5.3 Front



The front plate of the case holds all sensors. it has two holes to mount the I-Buttons and reserves a slot for the motion sensor.

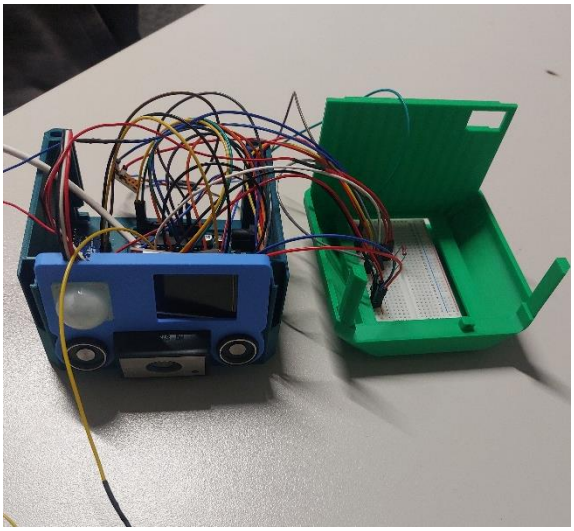
The TFT screen is fixed into place by pushing it into the 4 rods on the inside of the case.

The 3 smaller holes are used for the last I-Button sensor with led, it has screw holes so we used these screws not to only fix the sensor in place but also hold the front plate onto the rest of the case.

3.6 Final project

The following photos show the final project, featuring the case, sensors, and internal boards. The setup integrates all components seamlessly, ensuring full hardware functionality.

The screen is readable, and the I-Button sensors are easily accessible. The motion sensor is positioned at the front to detect movement in front of the device. The XYZ coordinates are displayed on the screen and reflect the device's position, updating dynamically when the device is moved or tipped.



4 Discussion

4.1 Code master device

When we began programming and testing the OneWire protocol, we encountered a problem. The OneWire library was not supported by the chip on the Arduino Nano 33 BLE Sense. After several weeks of troubleshooting, we discovered that the chip on the Nano was incompatible with the OneWire library. To overcome this issue, we switched to a different microcontroller, the Arduino Uno R3, which is compatible with the OneWire library.

4.2 Code slave device

The project requirements included reading embedded sensors on the microcontroller. Since the Arduino Uno R3 lacks built-in sensors, we had to incorporate the Arduino Nano 33 BLE Sense as a slave board. We used the I2C protocol to send sensor values from the Nano to the Uno.

During testing, we encountered an issue with receiving data. The data was being sent too quickly for the master device to process. After troubleshooting, we found that the master device's code couldn't keep up with the rapid data updates. To resolve this, we added a delay in sending the data, which ensured smooth and reliable communication between the boards.

4.3 Case

Overall, the case design process went well. In the first prototype, we overlooked the USB port for the Arduino Uno and the display was slightly off-center due to a small clearance gap. In the second version, we addressed and corrected all these issues, resulting in a functional and well-aligned design.

4.4 Setup

The setup of the physical hardware went smoothly overall. Installing the screen was straightforward due to its popularity and extensive documentation. However, the I-Button sensors presented a challenge as they have minimal documentation. To overcome this, we researched similar projects that used I-Button sensors and used their information. The installation of the other sensors also went well; despite limited documentation, we were able to figure them out quickly.