Jamie-Anthony Team 5
September 2023

LAB - Intro to nano IoT 33 and github II

Exercises

1 - Arduino Real Time Clock (RTC)

Setup the RTC ou your nano 33 IoT to show the same time as the lab time on the wall

Save your test as test_RTC.INO

Q How will you be able to use the RTC for your IoT device?

Timestamping Events: The RTC provides precise timekeeping, allowing us to timestamp events or data. This is valuable for various applications, such as logging sensor data, recording security camera footage, or tracking the timing of specific actions.

Synchronization: We can use the RTC to synchronize our IoT device with real-world time. This is particularly useful for devices that need to perform actions at specific times or intervals, like turning on lights, sending alerts, or managing schedules.

Event Triggering: We can set alarms or triggers based on specific times or intervals. For instance, we can use the RTC to trigger actions like sending notifications, capturing images from a security camera, or activating a smart door lock at a certain time each day.

Code:

```
void setup() {

 rtc.begin();

 // Set the RTC time to match the lab time on the wall

 rtc.setHours(4);

 rtc.setMinutes(35);

 rtc.setSeconds(0);

 rtc.setDay(18);

 rtc.setMonth(9);

 rtc.setYear(2023);

}

void loop() {

 int hours = rtc.getHours();

 int minutes = rtc.getMinutes();
 int seconds = rtc.getSeconds();

 Serial.print("Time: ");

 Serial.print(hours);

 Serial.print(":");

 Serial.print(minutes);

 Serial.print(":");

 Serial.println(seconds);

}
```

Q What happens when power is interrupted? You will need a button size lithium battery on the nano 3.3v pin to keep RTC alive after being put in sleep mode and BEFORE USB or VIN is disconnected:

When power is interrupted in an IoT device that relies on an RTC for timekeeping, there is a risk of losing the RTC's timekeeping data if it doesn't have a backup power source.

----

2 - Inertial Measurement Unit

Setup the IMU on your nano 33 IoT - Make the embedded LED light turn on when the board is held vertically

Save your test as test_imu_vertical.INO

Q - How will you be able to use the IMU for your IoT device?

Motion Sensing and Gesture Recognition: The IMU can detect changes in orientation and acceleration, allowing our IoT device to recognize gestures and movements. For example, you can detect when the device is shaken, tilted, or rotated and trigger specific actions or responses accordingly.

Stabilization and Image Capture: In applications involving cameras or image capture, the IMU can help stabilize images by compensating for device motion. This is particularly useful in drones and action cameras to ensure smooth and steady footage.

Code:

```
#include <Arduino_LSM6DS3.h>

void setup() {

 // Initialize the IMU

 if (!IMU.begin()) {

   while (1);
```

```
  }
  // Set the embedded LED as an output
  pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
  // Read accelerometer data
  float x, y, z;
  IMU.readAcceleration(x, y, z);
  // Check if the board is held vertically
  if (abs(x) < 0.1 && abs(y) < 0.1 && z > 9.0) {
    // If the board is held vertically, turn on the LED
    digitalWrite(LED_BUILTIN, HIGH);
  } else {
    // Otherwise, turn off the LED
    digitalWrite(LED_BUILTIN, LOW);
  }
}
```

----

3 - Arduino watchdog timer

Test the Watch Dog Timer

Save your test as test_WDT.INO

Q - How long should you set a WDT in your IoT deivce?

The appropriate Watchdog Timer (WDT) timeout value for our smart security system depends on several factors, including processing requirements, and its expected operational behavior. Here are some considerations to help you determine the WDT timeout value for a smart security camera:

Frame Rate and Video Processing: Consider the frame rate at which the camera captures and processes video frames. If your camera operates at a high frame rate or performs intensive video processing tasks, you may need a longer WDT timeout to ensure that it completes these tasks without resetting.

Resolution and Image Quality: Higher camera resolutions and image quality settings can require more processing time. Adjust the timeout to accommodate the time needed for capturing and processing images at the desired quality.

Motion Detection: If your security camera performs real-time motion detection, take into account the computational requirements of this task. A longer timeout may be necessary to allow sufficient time for motion analysis.

Network Communication: If the camera sends video or image data over a network, consider the time required for data transmission and any network-related processing. A longer timeout may be needed to ensure that data is transmitted successfully.

Code:

```
#include <avr/wdt.h>
void setup() {
 // Initialize serial communication for debugging (optional)
 Serial.begin(9600);

 // Enable the Watchdog Timer with a timeout of 2 seconds
 // You can choose a different timeout period as needed
 // WDTO_2S, WDTO_1S, WDTO_500MS, WDTO_250MS, WDTO_125MS, WDTO_15MS
 wdt_enable(WDTO_2S);
 // Additional setup code goes here
}
void loop() {
 // Your main loop code goes here
 // Ensure that your code executes in less than the Watchdog timeout period
 // Uncomment the following line to trigger a reset manually (for testing)
 // wdt_reset();
 // Print a message to indicate that the loop is running
 Serial.println("Loop is running...");
 // Simulate a delay to demonstrate Watchdog Timer reset
 delay(5000); // Simulate a 5-second delay (should trigger a reset)
}
```