Software Development for Real Time Embedded Systems- 605.715, Fall 2025

Matthew Halligan

Project 2

September 21, 2025

Professor Doug Ferguson

Project 2: Arduino Transmit Temperature Sensor Data

Project Requirements:

- Calibrate a temperature sensor connected to an Arduino. See “A Quick Guide to Temperature Sensors and Calibration.pdf” for some techniques.
- Using a Round Robbin with Interrupts design, get the Arduino to capture the
- temperature and convert to Fahrenheit.
- After the temperature has stabilized, start recording the Arduino temperature at a periodic rate of around 10s at room temperature, continue to record periodic temperatures after placing the Arduino in the refrigerator for 5 minutes, and continue to record periodic temperatures after removing from refrigerator and staying in the room for 5 minutes.
- Transmit the time and temp across a Serial bus like USB to your Host, others could be SPI or I2C if your Host had one of those busses.
- Export as a comma separated value file, read into a spreadsheet program and plot the temperature vs time.

Design:

Hardware list:

- Using an Arduino Uno,
- DHT11 Temperature/Humidity Sensor with breakout board,
- a bread board,
- three jumper wires

Configure the circuit according to Figure 1.

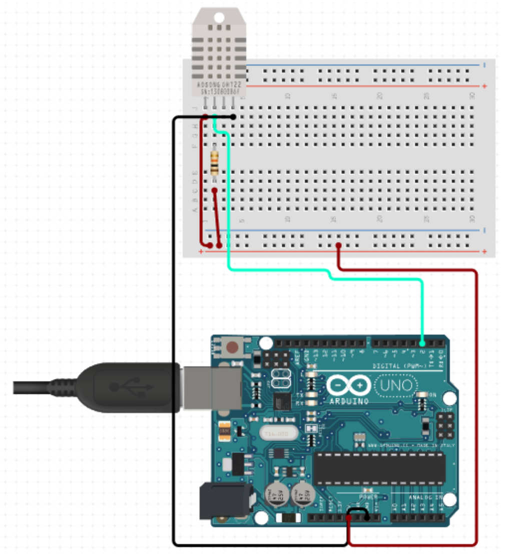

Figure 1 DHT11 Temp/Humidity sensor circuit diagram

Software Design:

The software design is influenced by the assignment instructions.

This firmware reads temperature and humidity from a DHT11 sensor at a fixed interval of 10 seconds, then when triggered via timer-based interrupt, it converts Celsius to Fahrenheit, and using serial export, prints CSV-formatted rows over the Serial port. Timekeeping is provided by Timer1 in CTC mode generating an interrupt roughly once per second. An ISR counts seconds and signals the main loop when it's time to sample. The main loop performs artificial tasking of producing a heartbeat suing a round robin design and performs light weight work such as serial printing a '.' character at 500 ms "heartbeat" and, when signaled, reads the sensor and prints one CSV-formatted row containing the ms count, seconds count, temperature in Fahrenheit, and humidity readings.

The software must sample the sensor every 10 seconds and completes this by implementing a round robin with interrupts design. There is no manual user interaction required once power and serial connection are supplied. The action of sampling and converting Celsius to Fahrenheit is considered high priority and atomic so it disables interrupts prior to entry and re-enables them upon return.

Figure 2 shows control flow of the program from setup of timers, timing registers, ISR vector selection, and main loop workflow.

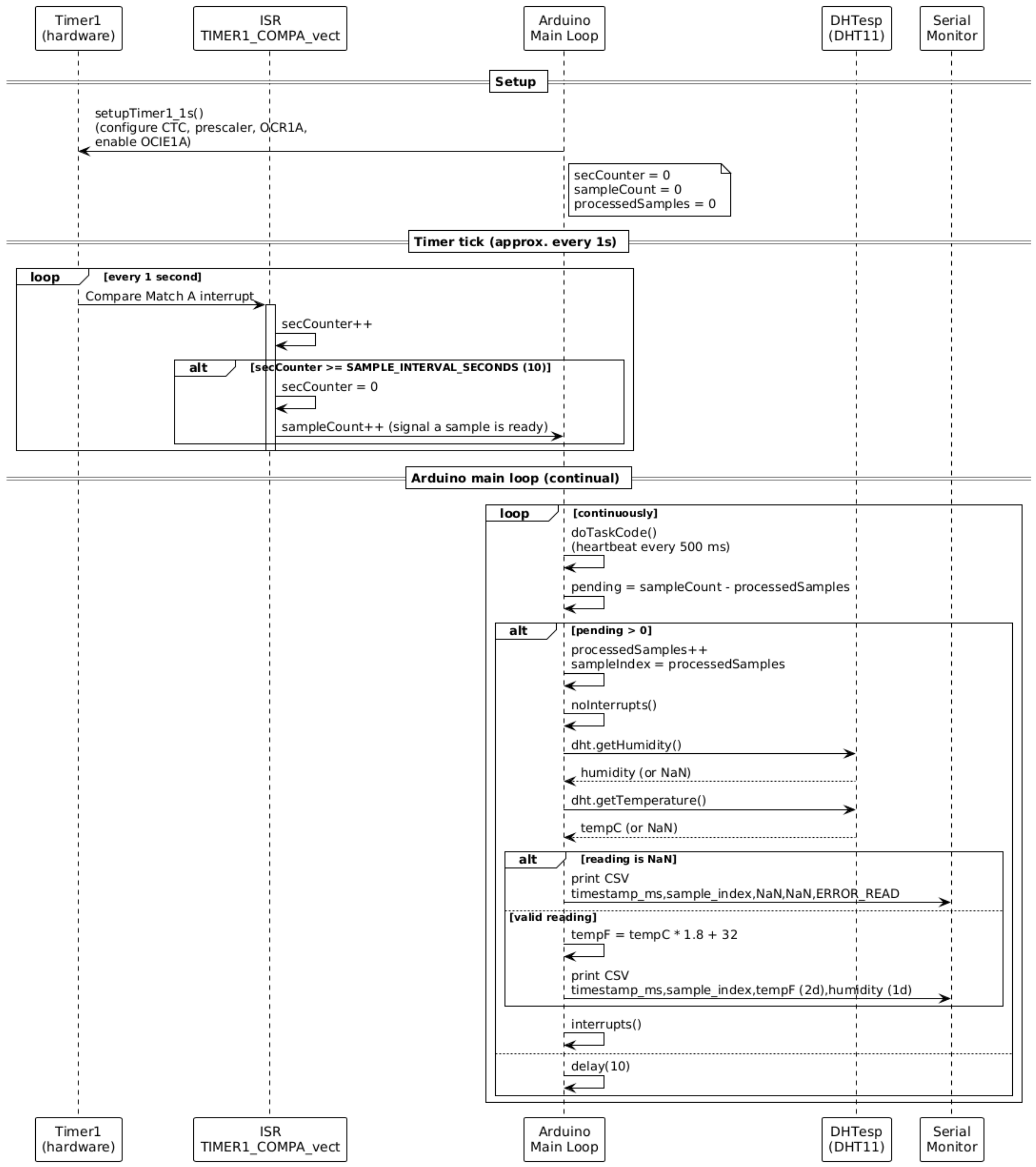
Timer1 (hardware)
ISR TIMER1_COMPA_vect
Arduino Main Loop
DHTesp (DHT11)
Serial Monitor
Setup
setupTimer1_1s() (configure CTC, prescaler, OCR1A, enable OCIE1A)
secCounter = 0
sampleCount = 0
processedSamples = 0
Timer tick (approx. every 1s)
loop [every 1 second]
Compare Match A interrupt
secCounter++
alt [secCounter >= SAMPLE_INTERVAL_SECONDS (10)]
secCounter = 0
sampleCount++ (signal a sample is ready)
Arduino main loop (continual)
loop [continuously]
doTaskCode() (heartbeat every 500 ms)
pending = sampleCount - processedSamples
alt [pending > 0]
processedSamples++
sampleIndex = processedSamples
noInterrupts()
dht.getHumidity()
humidity (or NaN)
dht.getTemperature()
tempC (or NaN)
alt [reading is NaN]
print CSV timestamp_ms,sample_index,NaN,NaN,ERROR_READ
[valid reading]
tempF = tempC * 1.8 + 32
print CSV timestamp_ms,sample_index,tempF (2d),humidity (1d)
interrupts()
delay(10)

Figure 2 Timer Setup, ISR Configuration, and Arduino Uno Main loop sequence diagram

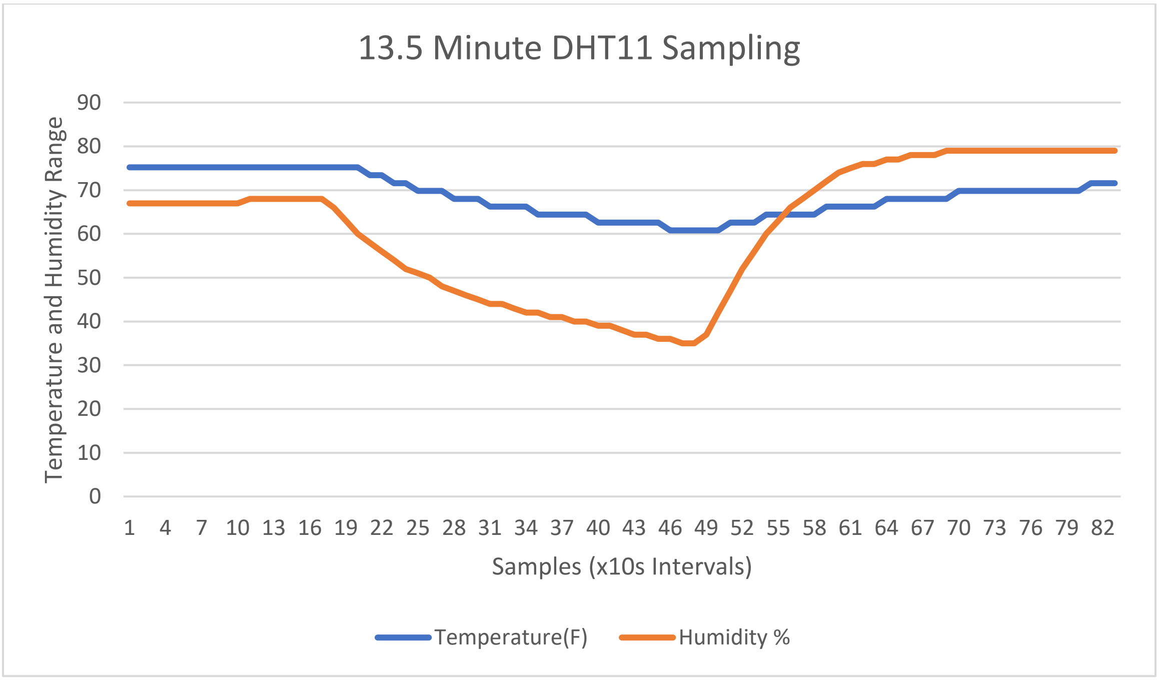

Figure 3 13.5 minutes of sampling DHT11, approximately 3 minutes to stabilize, 5 minutes in refrigerator, approximately 5 minutes stabilizing after refrigeration.

Figure 3 demonstrates approximately 13.5 minutes of sensor sampling data using the DHT11 temperature/humidity sensor. After approximately 3 minutes of stabilizing prior to the experiment, the sensor is inserted into the refrigerator where a steady decline is apparent around $x = 22$. During the experiment, it was raining outside so humidity was relatively high at slightly under 70%. Upon insertion into the refrigerator, humidity declines drastically. At approximately $x = 49$, the device was removed from the refrigerator and humidity is observed to raise drastically while the temperature slowly, but never fully returns to baseline temperature readings.

Video Demo Links:

Arduino + DHT11 Temperature/Humidity Sensing using Interrupt Driven Design Demo: https://livejohnshopkins-my.sharepoint.com/:v:/g/personal/mhallig2_jh_edu/Eb2s0XAH5LZKsJECO65dOBEBdEUPseBurNF4PCAW4s2vFA?e=gE6kUN

Project Code

```
//////////////////////////////////////////////////////
//
//  Matthew Halligan
//  Software Development for Real Time Embedded Systems, Fall 2025
//  Module 3/4 Project 2
//  Professor Doug Ferguson
//
// Read and export Fahrenheit-Converted Temperature Readings from DHT11 Sensor
// Implements Celsius to Fahrenheit conversion using timer based interrupts
// "Round Robin with interrupts Design" and serial export
//
// To use, ensure serial monitor is attached via Arduino IDE 2.3.6 set to 9600 baud
// No further action is required by the user once power and serial are supplied to Arduino Uno
//
//////////////////////////////////////////////////////

#include "DHTesp.h"
DHTesp dht;

const uint8_t DHT_PIN = 3;             // digital pin 3 for DHT data
const uint8_t SAMPLE_INTERVAL_SECONDS = 10; // sample every 10 seconds
const unsigned long HEARTBEAT_MS = 500; // task code background artificial interval

volatile unsigned long sampleCount = 0;      // number of samples requested by ISR
unsigned long processedSamples = 0;         // number of samples processed in main loop

volatile uint8_t secCounter = 0;            // seconds counter incremented by ISR

// --- Configure Timer1 for 1-second Clear Timer on Compare (CTC) ticks ---
void setupTimer1_1s() {
  cli();              // disable global interrupts while configuring
  TCCR1A = 0;
  TCCR1B = 0;
  TCNT1 = 0;
  // Clear Timer on Compare mode (WGM12 = 1)
  TCCR1B |= (1 << WGM12);
  // Prescaler 1024: CS12 = 1, CS11 = 0, CS10 = 1
  TCCR1B |= (1 << CS12) | (1 << CS10);
  // OCR1A for 1 second: ticks = F_CPU / prescaler = 16,000,000 / 1024 = 15625 -> OCR1A =
15624
  OCR1A = 15624;
  // Enable Timer1 compare A interrupt
  TIMSK1 |= (1 << OCIE1A);
  sei();              // enable interrupts
}

// Timer1 Compare Match A ISR at approx. once per second
ISR(TIMER1_COMPA_vect) {
  ++secCounter;
  if (secCounter >= SAMPLE_INTERVAL_SECONDS) {
    secCounter = 0;
    ++sampleCount;    // signal main loop to take one sample
  }
}

unsigned long lastHeartbeat = 0;
void doTaskCode() {
  unsigned long now = millis();
  if (now - lastHeartbeat >= HEARTBEAT_MS) {
    lastHeartbeat = now;
    // Serial.print(".");
  }
}

void setup() {
  Serial.begin(9600);
  delay(1000);

  // DHT11 initialization
  dht.setup(DHT_PIN, DHTesp::DHT11);

  // Timer setup
  setupTimer1_1s();

  // CSV header
  Serial.println("timestamp_ms,sample_index,temp_F,humidity_pct");
}

void takeOneSampleAndPrint(unsigned long sampleIndex) {
  float hum = dht.getHumidity();
  float tempC = dht.getTemperature();

  unsigned long ts = millis(); // ms time stamp since start

  if (isnan(tempC) || isnan(hum)) {
    Serial.print(ts); Serial.print(",");
    Serial.print(sampleIndex); Serial.print(",");
    Serial.print("NaN,NaN,ERROR_READ");
    Serial.println();
    return;
  }

  float tempF = tempC * 1.8f + 32.0f;

  // CSV: timestamp_ms,sample_index,temp_F,humidity_pct,event
  Serial.print(ts); Serial.print(",");
  Serial.print(sampleIndex); Serial.print(",");
  Serial.print(tempF, 2); Serial.print(",");
  Serial.print(hum, 1);
  Serial.println();
}

void loop() {
  // artificial task code that would otherwise be interrupted.
  doTaskCode();


  unsigned long pending = sampleCount - processedSamples;


  if (pending > 0) {
    ++processedSamples;
    unsigned long sampleIndex = processedSamples;
    noInterrupts();
    takeOneSampleAndPrint(sampleIndex);
    interrupts();

  }

  delay(10);
}
```