Research Project

Pearl, Rob, Emilie & Faisal
Professor Joy
COSC 365 - 001

Table of Contents

Introduction
(Emilie)

Literature Review
(Pearl)

Memory
(Emilie)

Operating System
(Rob)

Communication
(Pearl)

Conclusion
(Faisal)

References

Introduction

First created in 2005 at the Interactive Design institute in Ivrea, Italy. The Arduino is an open-source platform using easy-to-use software and hardware. Arduino boards can be implemented to read inputs given by sensors or a button and be processed by the requirements met by input data set by a microcontroller. The premise of the project is not having to manually turn on and off a humidifier once the humidity in the environment decreases to a certain amount(Humidifier Control, Arduino project Hub). The project consists of only a few components and supplies. In our project we will be using the Arduino Uno R3 board to fulfill the sensor and complete the output. The sensor that is being used is the DHT11 Temperature and humidity sensor w/built in resistor board. The Relay being used is the Tolako 5v Relay Module w/LED the LED will help the user identify that the humidifier and system is running. The humidifier is a generic outlet plug-in with an on and off switch. We will be utilizing the Arduino IDE and downloading the library of the DHT11 sensor in our system to input humidity data for the Arduino to read.

Literature Review

Digital Logic and circuit designing are basic features in both computer and electrical engineering fields, and it is critical that those concepts offer us a thorough understanding. While taking this course- Digital Logic, concepts that are introduced so far to help us with this project: Arithmetic of Numbers, Boolean Algebra, Karnaugh maps, circuits, digital logic gates and truth tables etc. Throughout the paper, four Frostburg University students: Rob, Emilie, Faisal and Pearl will explain an overview of using the Arduino in our circuit project. By doing this, we will be able to acquire the knowledge we need as well as the abilities and resolve to research & address challenging issues. With that in mind, this also demonstrates the significance of hands-on projects because the importance of hands-on learning allows us to experiment with trial & error, and learn from mistakes. Through our research, we learned about the capabilities and constraints of the utilization of the Arduino board.

Limitations & Benefits

Our study's main limitations were due to a lack of time, a time constraint, and limited equipment. Given how many classes and extracurricular activities we engage in, it has been difficult to set a certain meeting time. Regarding the equipment, it was a problem when we first started because the AdaFruit website didn't carry the Raspberry Pi and by the time, the Raspberry Pi would have been here, it would have passed that date the Fall semester ended. Another limitation was that due to the pricing and timing of the product - Raspberry Pi being sold out, the type of device that we utilized might influence the study outcomes because we had initially planned to use the raspberry pi. We ended up making changes by deleting unnecessary equipment, such as the AdaFruit breadboard and AdaFruit Featherwing OLED from our original

plan. The advantages we received from our research were the amount of easy, accessible information that was under the Arduino because there were a lot of very cool projects and knowledge for us to choose from and acquire inspiration for our study.

Research Questions

We were interested in the outcome of what the Arduino has to offer when it comes to generating an output to sense temperature and humidity as researchers in this study. Because of the daily adjustments, there was some concern about the amount of equipment required for this research to function. Fortunately, by doing so within a few days, we were able to create a comprehensive inventory of what was required to develop this project. Thus leading to our research question, “Will the Arduino make our Temperature & Humidity Sensor run more effectively and efficiently compared to the Raspberry Pi”?

Memory

The Arduino board uses three different memories in the microcontroller. The flash memory holds the sketch which is the program or uploaded code that the board runs. The second memory is the SRAM(static random access memory) ; it is the space where the sketch makes and uses variables when running. The last is the EEPROM where the programmer is able to use it as a long-term storage. The flash and EEPROM are the two memories that will retain as the power is turned off whereas the SRAM once the power is turned off the information will be lost. On the Arduino Uno R3 board it uses the ATmega328p chip containing 32KB of flash memory, 2KB of SRAM memory, and 1 KB EEPROM memory(Memory, Arduino). In our project we will be using the available memory provided by the Arduino to hold the data.

Operating System

As we are using an Arduino Uno R3, which is a microcontroller, instead of a Raspberry Pi, which is a microcomputer, it does not have a “normal” operating system. Instead of being able to run Linux or Windows, you can only choose a specific “scheduler OS” to install on the Arduino board. A scheduler (which is usually a lower-level part of an OS) is what determines how and when the different processes are *scheduled* to run on the processor.

Specifically there are two types of schedulers. Cooperative schedulers and preemptive schedulers. Cooperative schedulers don’t do a context switch (putting one process on hold and *switching* to another) unless the process at hand has been fully executed. Cooperative scheduling locks the processor into executing the process until it either terminates or finishes execution.
A preemptive scheduler is able to do a context switch at any time. This allows the processor to focus on several different processes at once, working on bits and pieces at a time.

There are several popular Scheduler OSes available for the Arduino Uno R3, we will list some of the most popular ones.
The default “OS” that comes with Arduino is neither cooperative or preemptive, and thus cannot technically multitask. All the default firmware does is execute each process, one after another. It acts more like firmware than an OS, which makes it very simple and lightweight.
Moving onto 3rd party schedulers, we will start with the lightest and move up to the most complex.

CoopThreads is a very lightweight, open-source scheduler that will run on hardware as light as even 8-bit microcontrollers. It uses the “Round Robin” method of scheduling. Round Robin scheduling assigns equal time to each process without prioritizing one over another. It acts in a “circular” manner, assigning tasks for a certain amount of time, stopping the task, and moving the current task to the end of the queue.

FreeRTOS is an open-source scheduler, available for all microcontrollers that use the AVR standard (most of Arduino’s boards do). FreeRTOS can be configured to run in either cooperative *or* preemptive scheduling mode. This allows for much more flexibility over schedulers that can only operate one way or another.

Simba is also open source, which like FreeRTOS can run in either cooperative or preemptive scheduling mode. However, it is also more versatile than the other options in this list. Simba can have a basic file system, networking features (with the various internet connectivity protocols such as TCP, UDP, and HTTP), and can even do pseudo multi-threading.

Comparing these, CoopThreads is lightweight, able to run on even 8-bit microcontrollers. However, we don’t need an OS this light for our board. The Round Robin method of scheduling also adds a layer of complexity that we don’t need for our project.

FreeRTOS runs on almost every board that Arduino produces, and can be configured to either scheduling mode, like Simba. However, Simba is far more versatile than FreeRTOS. Simba's extreme versatility and configurability make it by far the most useful choice out of the most popular options.

Choosing a scheduler OS to use on the Arduino really depends on what you need it for. For our project, we don't need any of the fancy features of these scheduler OSes. Our project will not require an OS outside of the one that comes installed on the Arduino itself.

Since the "OS" we'll be using comes installed on the Arduino by default, we will not need to install anything. However, for the sake of thoroughness, we will detail the installation of Simba here.

You can either use Arduino's IDE or a dedicated PlatformIO installer, but we will use the PlatformIO IDE for simplicities sake.

This guide is summarized from the guide at

https://simba-os.readthedocs.io/en/latest/getting-started.html

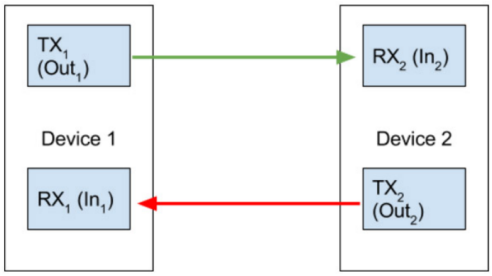
1. Open PlatformIO, select *project-Examples*, and *Simba/blink*
2. Upload the OS by clicking the upload button

You can also install Simba using Arduino's IDE, but it is much more complex.

Communication

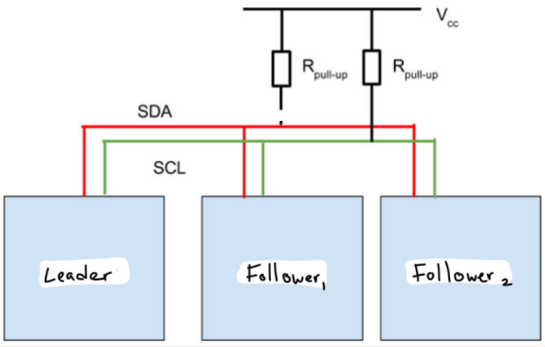
The Arduino can communicate in a variety of ways and under a variety of conditions, but it always does so in a serial manner. It is a concept, but how the Arduino board communicates with a computer or other electrical devices involves using the uno, nano, micro, and mega pins of Ones and Zeros. In a digital signal, data is transferred in a sequence of two ways: Highs and Lows and Lows to Highs. Fun fact there are three different protocols that are utilized in device communication that also affect the Arduino:

UART Protocol



UART is a form of serial communication because data is transmitted as a sequential bit. The term UART refers to the onboard hardware that manages the packaging and translation of serial data. Which is where the hardware comes into play because that's the only way the UART protocol can communicate. On the Arduino Uno, there is one serial port dedicated for communication with the computer the Arduino is connected to.

SPI Communication



(SPI) *Serial Peripheral Interface* Follows a leader-follower model meaning that the communication relies on defining one device as a leader, and other devices as followers . When it comes to the implementations on the Arduino, the Arduino is the Leader

device (SPI.h). The SPI digital pin connections for SCK, MOSI, and MISO are predefined on Arduino boards which are considered the followers.

I²C Communication

I2C Inter-integrated circuit is the last line of communications. Although the circuit is the most complicated out of the three protocols, it addresses the drawbacks from the last two and corrects those issues by connecting multiple leaders to multiple followers and simplicity - requiring only two wires and resistor.

Conclusion

In conclusion, with the information gathered on the utilization of the Arduino and the lessons we learned throughout Digital Logic class. We are confident that our research will be successful because of the numerous outstanding features and advantages the Arduino board offers. With the installation of the three different memories, the information that is gathered from the temperature of the humidifier will make storing data very easier to hold. Significantly due to the fact it is a microcontroller; One we do not need to worry about having the circuit run through the computer application software: Linux, iMac or Windows because it can easily be installed through an OS scheduler. And two, throughout that Arduino's OS system, it'll not only make the communication between other devices run very smoothly but give us the chance to experience our research study come to life!

References

"Arduino - Communication." *Www.tutorialspoint.com*, www.tutorialspoint.com/arduino/arduino_communication.htm. Accessed 6 Oct. 2022.

"Arduino Boards | Types and Uses of Arduino Board." *EDUCBA*, 1 Dec. 2020, www.educba.com/arduino-boards/. Accessed 6 Oct. 2022.

"Arduino Humidifier Control." *Arduino Project Hub*, create.arduino.cc/projecthub/gatoninja236/arduino-humidifier-control-2e7805.

"Arduino Operating System: Best Options of 2021." *All3DP*, 14 Feb. 2021, all3dp.com/2/best-arduino-operating-system/. Accessed 6 Oct. 2022.

"Communication between Arduino UNO." *Arduino Project Hub*, create.arduino.cc/projecthub/masteruan/communication-between-arduino-uno-c1caa5. Accessed 6 Oct. 2022.

"FreeRTOS - Market Leading RTOS (Real Time Operating System) for Embedded Systems with Internet of Things Extensions." *Freertos.org*, 2010, www.freertos.org/.

"Getting Started — Simba Master Documentation." *Simba-Os.readthedocs.io*, simba-os.readthedocs.io/en/latest/getting-started.html. Accessed 6 Oct. 2022.

"History of Arduino Boards." *AHIRLABS*, 11 Oct. 2017, www.ahirlabs.com/2017/10/11/history-of-arduino/. Accessed 6 Oct. 2022.

Iyer, Rahul. "Arduino Serial Tutorial - Arduino Communication Protocols." *Device Plus*, 29 Nov. 2016, www.deviceplus.com/arduino/arduino-communication-protocols-tutorial/.

"Memory." *Www.arduino.cc*, www.arduino.cc/en/Tutorial/Foundations/Memory. Accessed 6 Oct. 2022.

Moqvist, Erik. “Eerimoq/Simba.” *GitHub*, 16 Aug. 2022, github.com/eerimoq/simba. Accessed 6 Oct. 2022.

Peterson, Manny. “Overview.” *GitHub*, 1 Oct. 2022, github.com/MannyPeterson/HeliOS. Accessed 6 Oct. 2022.

Stolarz, Piotr. “CoopThreads.” *GitHub*, 19 May 2022, github.com/pstolarz/CoopThreads. Accessed 6 Oct. 2022.

“Welcome to Simba’s Documentation! — Simba Master Documentation.” *Simba-Os.readthedocs.io*, simba-os.readthedocs.io/en/latest/index.html. Accessed 6 Oct. 2022.