Introduction

September 25, 2024 8:34 PM

Note from Tim: Based on the instructions I understand we have a certain freedom to our blog. I've decided to structure based off of OneNote notebooks - I absolutely love OneNote and live by it professionally. Each unit will be a "Section", then sections will be broken into pages for introductions/descriptions of the unit, as well as a personal summary of readings, and SIK exercises. For each unit submission, I will do an extract of that unit to HTML or PDF. Final project submission will be a complete export of the notebook. I will include images/videos examples of exercises within the pages when possible, however they are also available via my git repo: https://github.com/malcolmsdad/COMP444/tree/main (FYI Malcolm is the name of one of my adorable dumb cats)

Learning Objectives

After completing this unit, you should be able to

- define the term *robot* and provide a brief history of robots and robotics.
- list the components of a typical robot.
- continue on with more advanced robotic material.

Readings

Please read the following chapters in the textbook:

- What Is A Robot? (Chapter 1)
- Where Do Robots Come From? (Chapter 2)
- What's in a Robot? (Chapter 3)

Questions to Ponder

At the end of each chapter in the assigned readings there are questions labelled "Food for Thought." Please answer these questions as best you can in your weblog, which will become part of your portfolio of competence submitted for marking during this course.

Exercises

Exercises for this unit can be found in the *Instructor's Weblog* on the Landing. Please follow along with the exercises and programs using your own Arduino and the SparkFun Inventor's Kit, and keep a record of your explorations in your own weblog.

Further Readings

At the end of each chapter in the assigned readings you will find a section titled "Looking for More." While the links and readings mentioned in this section are not assigned, please feel free to examine them if you are interested, or ask questions about them on the Landing.
Bookmark the brief history of the Arduino found on *Wikipedia* http://en.wikipedia.org/wiki/Arduino, and check out the official Arduino home page at http://www.arduino.cc/.

Instructor Blog Entry

October 9, 2024 11:21 PM

As mentioned in the Study guide and this notebook, the companion text workbook is somewhat incomplete, and does not completely apply to our chosen hardware platform. The workbook also does not correspond well to the SIK Guide, in that the guide explores some hardware elements in different order than the text's workbook. However, this is not really a significant difficulty, as we want to first explore the various hardware elements in our Inventor's kit, after which we can tackle some of the text's robotic questions.

Unit 1 is a general discussion of locomotion. Unit 2 will further explore robotic locomotion, but for now we shall begin by working through the SIK tutorials.

Bearing that in mind, you should begin to work through the SIK Guide tutorials in Section 2, making notes of your progress, questions, challenges and solutions as you go. Due to the structure of the SIK Guide, I recommend you tackle the circuits and programs in the order they are presented, starting with Circuit #2 and continuing through Circuit #7. These circuits comprise the tutorial's coverage of sensor inputs, which will be put to further use in Unit 4, and pave the way for the next unit's circuits.

Activities using the text Companion Workbook: Read through the text companion workbook section titled Robotic Components. Answer the questions posed in this section of the workbook in your own weblog.

From <https://landing.athabascau.ca/pages/view/242993/unit-1>

Chapter 1 - Robots

September 25, 2024 8:36 PM

Well, here we go. The beginning of another course. I'm very much looking forward to this one though, this is much more aligned with my personal interests and hobbies.

I've been programming nearly my entire life, professionally for the last 20 years. I've also dabbled with embedded and micro-computing quite a bit as well. I'm an avid Raspberry PI hacker, have done a lot of projects with Arduinos, ESP32s and even Intel Galileo. Robotics is totally new to me though, so I'm really excited to see dive deeper into that.

Chapter 1: Robotics

This chapter provided a very brief history of robots, the original inventor of the term (Josef Capek, or his brother). It also described the evolution of the term robot, as well as defining the modern meaning of the word. (See Definitions section)
Primary concepts of robots are introduced such as the difference between teleoperation and autonomy. A great deal of emphasis was put into defining sensors and robot must sense its environment to be considered robotic.

Definitions

October 7, 2024 10:21 PM

Robot: an autonomous systems that exists in the real world. It can sense its environment and interact with it. It must be an autonomous system, that exists in a physical world, can sense its environment and act upon it in order to achieve some goals.

Autonomous: an actor capable of acting on the basis of its own decisions and is not controlled by a human.

Teleoperate: to operate a system from afar.

Sensors: Sensors are physical devices that enable a robot to perceive its physical environment in order to gain information

Embodiment: having a real body

Niche: environment or position in ecosystem

Food for Thought

October 9, 2024 5:24 PM

What else can you do from afar, by means of teleoperation? You can talk, walk and see, as in telephone, telegraph and television. There is more. Can you think of it?

The general theme for all of these question is: are they robots. In order to properly answer these questions, I believe it'd be best to review the definition and requirements to be declared a robot.

The definition of a robot is an automated machine that can detect is environment, and interact with it. It also most meet the following requirements:

- Be autonomous (not controller by human)
- Exist in physical world
- Sense its environment
- Can act upon its environment
- Achieve some goals

Is a thermostat a robot?

Yes and No. Like robots, thermostats can vary wildly.

Modern thermostats are intelligent, loaded with sensors, connected to the internet, paired with mobile apps and easy to program - a far cry from the days before IoT. The latest in thermostat technology sees the devices optimizing home energy use and comfort by through data collection and analysis such as downloading weather patterns to predict heating or cooling requirements, monitoring room or house occupancy to reduce energy use when not needed. Learning 'time to heat' statistics to predict when heating or cooling cycles should begin. In these situations, I would consider thermostats robots because:

- Some newer thermostats have the ability to learn users patterns and temperatures and adapt heating/cooling cycles based on those patters - this demonstrates autonomy
- Thermostats absolutely exist in the real world
- Thermostats physical capability is the manipulation of electronical currents to one or many devices such as HVAC appliances, such as fans, humidifiers, furnaces, boilers, air conditions, heat pumps etc. The goal of this action is to control the devices environment.
- The goal of these modern thermostats is to increase the comfort (temperature, humidity) and reduce energy usage.

Meanwhile, older thermostats, which consist of a simple dial, or basic display and temperature control button, are not robots. These thermostats rely on solid state or mechanical process to trigger heating/cooling cycles. These thermostats rely more on physics than robots. Comparing against the list earlier, we can see older thermostats:

- Exist in the physical world
- Can act up on its environment

But that's about the limit. Older thermostats:

- Do not sense their environment, although it may seem that they do, since they need to know the temperature of the environment, there is no perception in the thermostat when the furnace turns on due to cold air triggers - it's simply a thermodynamic reaction
- They have no goals to achieve, they simply operate

Is a toaster a robot?

A toaster is definitely not a robot. Toaster do not sense their environment and act upon it. They simply active heater coils for a user input amount of time.

Some intelligent programs, also called software agents, such as web crawlers, are called "soft bots" Are they robots?

No, these are not robots, because they do not satisfy the condition of "existing within the real physical world"

Is HAL, from the movie 2001, the Space Odyssey a robot?

Similarly, HAL is not a robot, because it does not satisfy the condition of "existing within the real physical world"

Chapter 2

October 9, 2024 10:05 PM

I appreciate the short and to the point chapter 1. Chapter 2 discusses more history of robotics.
The first topics is "Control Theory" which is defined as the mathematical analysis of automated control systems. I wasn't satisfied with this definition, so I did some secondary research. My initial thought was it was a specific theory describing control system, but in fact it's much more broad; its an entire branch of applied mathematics that performs objective analysis of feedbacks to evaluate and optimize system responses to goals and inputs.

The chapter then discusses cybernetics which describes the study of biological organism and comparison against artificial systems, a perfect prelude into the next section about a series of biomimetic tortoises by Will Grey Walter.
These robots, named Elmer and Elsie, were early tricycle based robots, that were able to sense and navigate through their environment, and even follow basic instructions based on light or sound inputs.

These tortoises were used to introduce concepts including: analog, analog electronic circuit, reactive control, emergent behaviour and artificial life. The analog and analog circuitry are now much outdated, due to the invention of the transistor. Reactive control is a new concept I learned, and something I'm quite concerned with, because it does focus heavily in a robot I'm pondering for the final project. I'm looking forward to Chapter 14 to go in more depth. Reactive control, led to an interested observed phenomenon called emergent behaviour which meant how robots exemplified animal like, realistic behaviour.

The chapter continues with another evolution of the tortoises; Valentino Braitenberg, wrote of a series of thought experiments (gedanken experiments) who described a series of minimalist robots demonstrative an inhibitory connection between sensor inputs and the robots reaction could mimic social behaviour, and even love or aggression. He did this by introducing concepts of his robots (Braitenberg vehicles) light sensors (photophilic and photophobic) reaction to light in positive (excitatory) or negative (inhibitory) connections.

The study to these emergent behaviours and input/output connections led to Artificial Intelligence, which is the next section. AI was officially born in 1956, Dartmouth University in Hannover, NH, USA. Defined AI as requiring:

- Internal models of world
- Search through possible solutions
- Planning and reasoning to solve problems
- Symbolic representation of information
- Hierarchical system organization
- Sequential program execution

AI is also something I'm very much interested in and look forward to further reading.
The chapter then described some early AI implementations in robots including Shakey, Flakey and Hillaire, CART and Rover.
Finally, the concept "types of robotic control" is introduced, which describe the different directions of robotic study including reactive control, behaviour based control and now purely deliberative.

Definitions

October 14, 2024 9:45 AM

Control Theory: A mathematical analysis to automation control systems

Cybernetics: a field of study that analysis biological systems at the nerve cell (neuron) level. It studies and compares biological and artificial systems with the goal of finding common properties or principals between the two.

Biomimetic: Machines with properties similar to biological systems or imitate biological systems in some way

Braitenberg Vehicle: a simple robot initially equipped with a light sensor and servo and programmed to react in a variety of ways based on light inputs. These robots were supplemented with additional sensors to study their reactive behaviours.

Photophilic: Act of being attracted to light

Photophobic: Act of being afraid of light

Excitatory Connection: The stronger the input, the strong the motor output.

Inhibitory Connection: The stronger the input, the weaker the output.

Artificial Intelligence: field of study that defines intelligence in machines and what is required for a machine to be intelligent. Understanding of world, ability to research, plan and reason solution. Ability to represent information symbolically, have hierarchical system organization and execute programs sequentially.

Shakey: An early intelligence robot that lived in a carefully controlled environment, with limited sensory inputs, and given a task. The robot would then attempt to solve the task iteratively through trial and error.

Vision-Based Navigation: Navigation of a robot through it's environment using vision sensors. Initially demonstrated in the CART robot, it was very slow to react due to time to process visual information.

Types of Robot Control: Different approaches to robot control including reactive, hybrid and behaviour based controls.

Food for Thought

October 9, 2024 7:37 PM

How important is it for robots to be inspired by biological systems? Some people argue that biology is our best and only model for robotics. Others say that biology teaches us valuable lessons, but that engineering can come up with its own, different solutions. The airplanes is a popular example of non-biomimetic engineering; people's first attempts to build flying machines were inspired by birds, but today's planes and helicopters share little in common with birds. Does it matter what kind of robot you are building (biomimetic or not)? Does it matter if it is going to interact with people? For more on this last topic, wait for Chapter 22.

I disagree with robots being inspired by biological systems, both for defining or modelling artificial intelligence and form, or physical attributes. We hardly understand biology to any point that we are able to reproduce to its most basic element. Only last week did we (humans) finally map the brain of a fruit fly, if the penultimate robot is a bipedal humanoid (like demonstrated in some many movies and electronics shows), then that'd require understanding the biology of much more complex biology like humans.
Even on our tiny little planet, biology of creatures and regions ranges wildly. Some creatures rely upon hands and fingers, others use claws, or wings, some even use suction cups and beaks to interact with things - who is to say that some new form of mechanical interaction may occur. Perhaps humans could create a method of manipulating an object with gravitational waves, or ionic thrust. Those methods of interaction could be far superior to a couple fingers.
Robotics should not limited to what biology has evolved to offer, it should be design ed and implemented to best suits its goals or needs.

Chapter 3

October 9, 2024 10:05 PM

The chapter starts with a refresh on the definition of a robot. It then describes the main components; a physical body, sensors, efforts and actuators, and a controller. The chapter then goes into further detail of physical bodies, called embodiment and their limitations; including comparing human limitations to robot limitations.

Sensing is then examined, including describing sensor perception, storage of sensor information (called state), how that information is perceived (observably, partially, or hidden) as well as how often that information is categorized (discrete or continuous). Storage techniques and requirements are then described - in quite detail.

The chapter then moves onto the more physical properties such as Actions, Brains and Brawn. Here we learn how robots can act upon their environment using effectors and actuators. The types of activities are broken into two categories: Locomotion or manipulation. The concept of degrees of freedom is introduced as the limitation in which a manipulator can move.
The chapter used limitation as a prelude into the next section which described power limitations imposed on robots, and how power is a major problem in practice robots, including storage, isolation, performance and replenishing.

Finally the chapter describes the controllers importance to a robot and how they are the 'brains of the robot'. Controllers use hardware and software programming to use sensor input to decide on how what to do.
Pseudocode is introduced as the method the chapter will describe programming of a robot, and autonomy is introduced as the ability to make ones own decision and act upon them.

Definitions

October 14, 2024 10:28 AM

Embodiment: the act of having a physical body

Sensors: physical devices that enable a robot to perceive its physical environment

Sensing Perception: The process of receiving information about the world through sensors

Niche: Environments that animals or robots live or operate within.

State: The robots description of itself at any point in time.

Observable/Partially/Hidden State: a state may be observable, partially observable or hidden to a robot.

Discrete / continuous state: Discrete information is up or down, blue or red, whereas continuous is information such as miles per hour.

State space All possible states a system can be in; ie: a light switch is on or off

Space: Refers to all possible values or variations of something

External State: The state of the world as the robot can perceive is the external state.

Internal State: The state of the robot as the robot can perceive it.

Representation/Internal Model: all information relating to the world collected by sensors

Sensor/Perceptual Space: The space required for all a robots sensors, in every possible sensory reading or permutation.

Effectors: Devices that enable a robot to take action, do physical things. Ie: Flippers, wings

Actuators: Underlying mechanisms that do actual work for a robot such as motors or muscles.

Degrees of freedom: The amount of movement a manipulator can move.

Locomotion: The act of moving around, going places

Manipulation: The act of handling of objects

Pseudocode is an intuitive way to describe the controller without using any particular programming language.

Autonomy is the ability to make ones own decision and act upon them.

Food for Thought

October 9, 2024 7:52 PM

What do you think is more difficult, manipulation or mobility?

Think about how those abilities develop in babies, children, and then adults.

- How large do you think your sensor space is?
- Can you think of things or information in your life that is observable, partially observable, or hidden?

I believe that manipulation is a more difficult task to achieve for a robot. If we consider what the most common form of mobility for machine is based on wheels and circular motion. Circular motion is easily accomplished by the most basic motor. A set of wheels, a simple motor and a bump sensor and you can easily achieve mobility.
Manipulation however is much more difficult. In order to manipulate an object, the robot must first sense or perceive the object, its location and orientation, as well as how to manipulate it. The robot also required effector and actuators to manipulate with the object. The number of required components in order to manipulate in object is much more

SIK Exercises

October 12, 2024 10:52 PM

Recommended and individual SIK exercises will be listed within this section. (The first two excercises are part of Unit 0, however their code can also be linked from my git repo)

Experiences during the projects, and appropriate visual aids will be used.

Any illustrative material that cannot be pasted within my blog will be available at (typically GIFs): https://github.com/malcolmsdad/COMP444/tree/318fbd46aea4ad804a20a212c5b05cf5c43c1bab/Excercises

NOTE: I have over 20 years professional programming experience, and been dabbling my entire life. I will not remark on theories, practices, syntax or methods that I have already used in my past - I will however, be completely open and transparent with my exercises including mentioning silly or careless mistakes. Silly 'noob mistakes' like missing semi-colon, or an incorrect casing can happen to any developer and actually cause a lot of headaches. I feel dealing with even this most basic mistakes important practice to catch and fix.

Circuit 1A: Blinking LED

October 14, 2024 10:58 AM

(Copied from Unit 0)

I connected followed the instructions in the SIK manual for Project 1A. It was a very simple setup, I sure enjoyed the breadboard - redboard place holder, nice touch by sparkfun Electronics. Setting up the board, LED, resistors, jumpers, USB and IDE was pretty straightforward, something I've done a dozen times before. To my surprise though, the program did NOT run the first time. I immediately knew the LED was in backwards, unplugged the board, switched the LED, and boom, problem solved - a nice blinking light.

Now, I wonder what else I can do with this? How about making the light blink S.O.S.? In order to do that, I'll need the code to make a light blink, an LED, and the timings of when the light should blink on and off. I've already got 2 of 3 requirements, a quick google and I've found that: A dash should be 3 units of length, a dot is 1 unit of length, the time between letters is 1 unit of length, and time between words is 3 units of length. Easy enough to encode those variables and make some functions to represent a dot or dash. It's up to me what a unit of measure is, and my first guess was clearly wrong:

(Seizure warning - Flashing light!)
P1A- Blink SOS - too fast (seizure warning!)

But after a quick tweak of one variables, we've got a great looking S.O.S. flashing light (I'll post the code at the bottom of this posting)
https://youtu.be/VuwL4KKGUFk

That about wraps up the first entry, and project. Was a lot of fun and looking forward to doing more. As for that S.O.S. program, it sure would be neat to do the opposite project; switch the LED for a button, and get the Arduino to read button presses into Morse code - maybe another time.

Challenge 1 Git source code:
https://github.com/malcolmsdad/COMP444/tree/ec3001ec11c7f952c27cc67c8e259b25fa814379/Unit%200/C1A%20Blinked%20LED%20-%20SOS%20Blink

Challenge 2: Constant LED:
https://github.com/malcolmsdad/COMP444/tree/ec3001ec11c7f952c27cc67c8e259b25fa814379/Unit%200/C1A%20Blinked%20LED%20-%20Constant%20Blink/SIK_Circuit_1A_BlinkingLED_Constant

I then attempted the constant LED challenge and noted that approximately 12-13 millisecond delay was the limit for when the LED appeared to be constantly lit.
Clock cycles: Just out of curiousity I googled the datasheet for the Redboard to see if its clock cycle had any impact on my test. According to the datasheet the clock cycle is anywhere from 10 - 40 MHZ. According to UnitJuggler the upper limit, 40 MHZ, converts to 25 nanoseconds, and given 1 nanosecond is equal to 1 million milliseconds, there would be no impact.

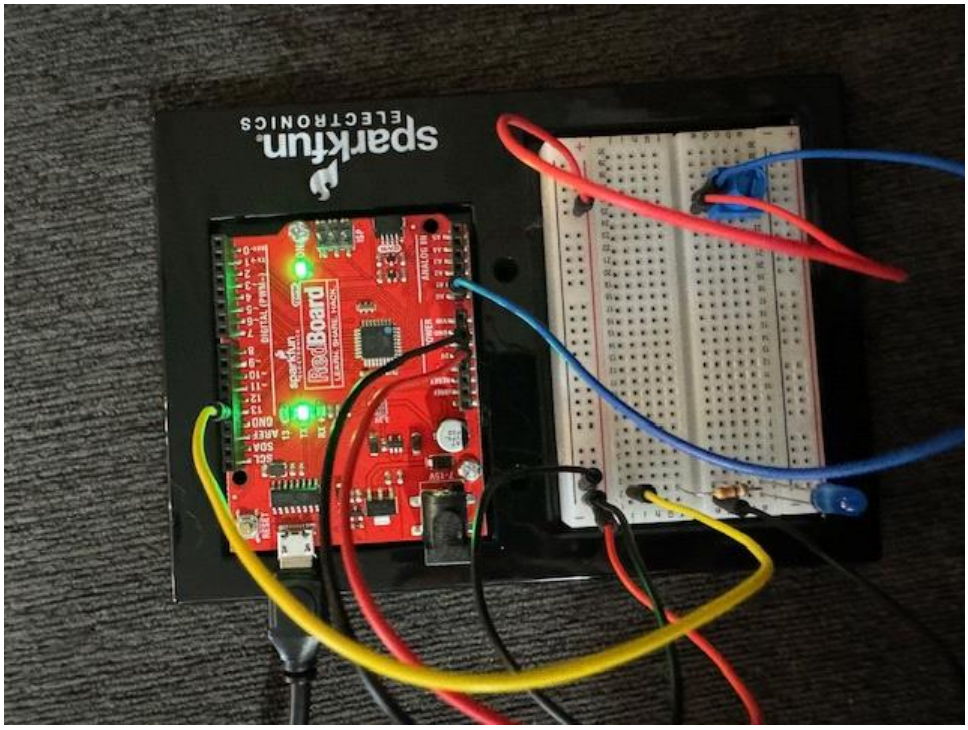
Circuit 1B: Potentiometer

October 13, 2024 11:07 AM

Circuit 1bB: Potentiometer
This task was not listed as part of the exercise list, but I wanted to give it a shot as extra practice. I have had experience using potentiometers before. During covid, I found myself diving into several projects that I'd always wanted to try, including building a custom full size arcade emulator cabinet. This included Raspberry PI, dozens of arcade switches, USB encoder and an old heavyduty steering wheel. That steering wheel had no native drivers for linux, so I had to read the output of the potentiometer of the steeringwheel, and pedals, then translated them into respective degrees of turning, or throttle positions. The arcade was a total success, and currently I'm testing various infrared, camera based 'light guns' - lots of fun! Onto the actual exercise:

Attempt #1: Fail
Good thing too, the first attempt at running the application failed. I set up the wires as described in the instruction book, then uploaded "ReadAnalogVoltage" program, but nothing happened!

No LED:

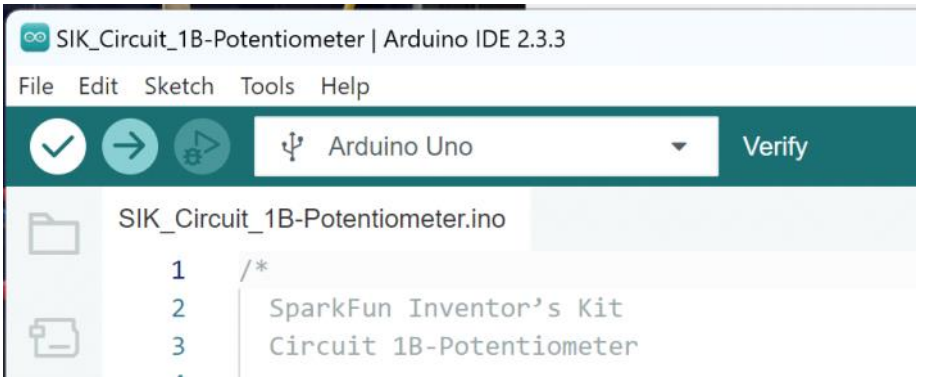


Attempt #2 / Big Whoops
I double checked the wiring, and I must remark, my eyes are not what they used to be! I have to use a lighted magnifying to help me read the labels on the board and textbook. Pulled and rewired the potentiometer and LED, then re-uploaded. Still no flashing light! This time, I opened the serial plotter and confirmed that the values are being red from the potentiometer sensor. At this point I'm really confused, I can't find anything wrong with the wiring at all, lets check the code. Aha. Wow did I goof. The code I'm running doesn't interact with an LED at all, it simply reads the potentiometer and console outputs the values - I'm running the wrong code!

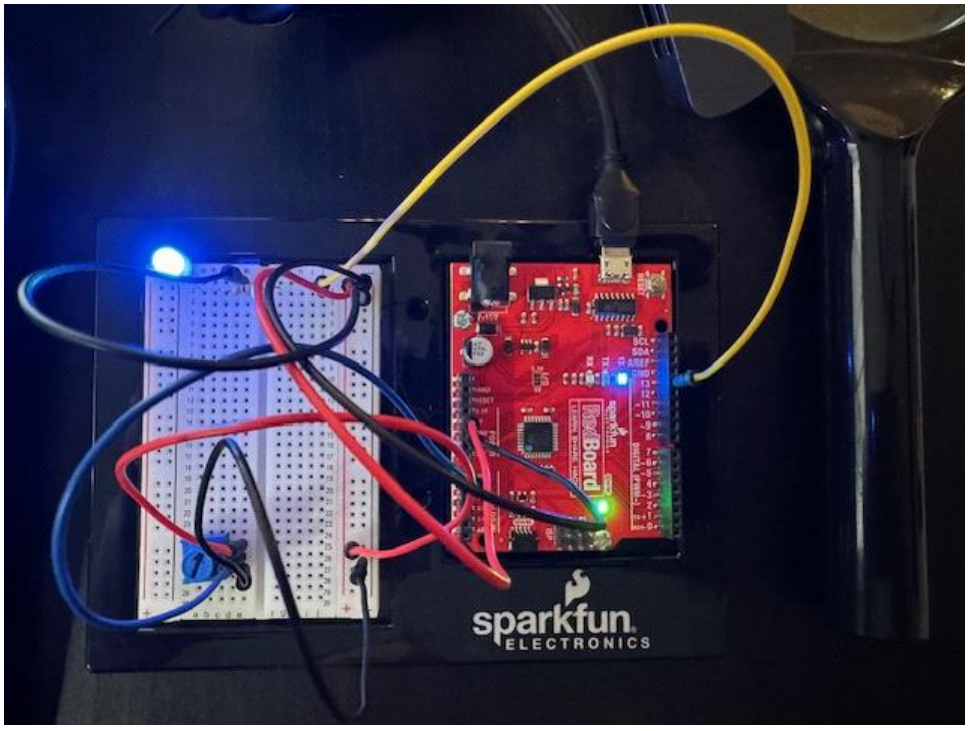
Wrong code:

Attempt #3: Success
Uploading and running the correct code resolved the problem perfectly - the program ran perfect;
LED responded to potentiometer as expected.

Correct code:



Flashing blue LED:


sparkfun
ELECTRONICS

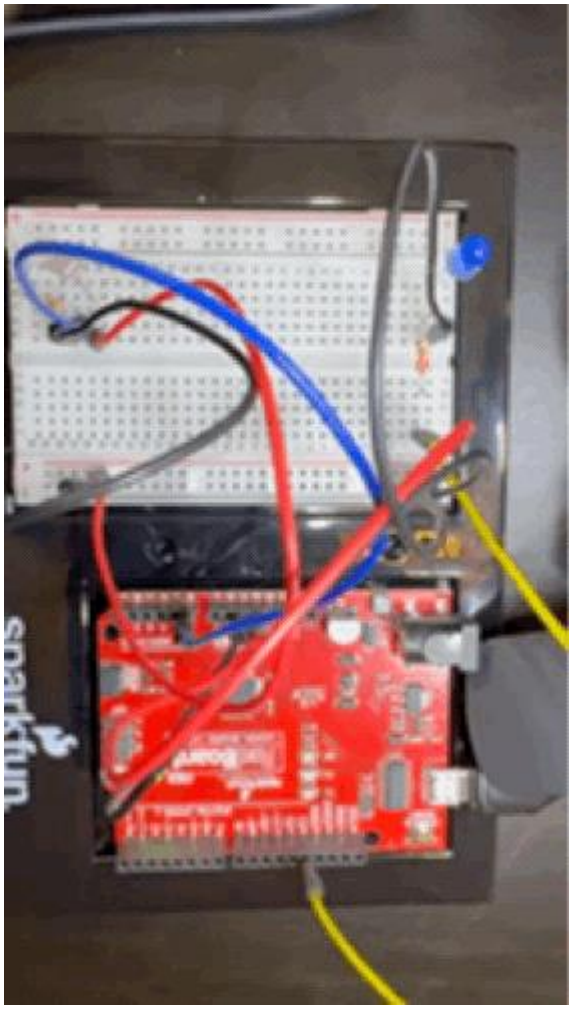
Circuit 1C: Photoresistor

October 13, 2024 11:19 AM

Time to learn how a photoresistor works.
A photoresistor changes the resistances based on how much light it receives.
Similar to potentiometer, photoresistors work by dividing voltages.
This practice introduced the concept of analog to digital conversion, as well as voltage dividing using resistance / voltage calculation.

Setup:
This was an easy one, it was very similar to the previous exercise 1b, with a swap of photoresistor for the potentiometer, and add a resistor.

Testing:
Initial test appeared to be a failure. The LED would not change when I moved my finger over the sensor. Upon reviewing the program instruction sheet, I noted that the some dim rooms may be too low for the threshold. I turned on another light, and immediately the resistor worked as expected - Success!



Project 1C
Photoresi...
(Animated GIF of LED reacting to finger)

Circuit 1D: Nightlight

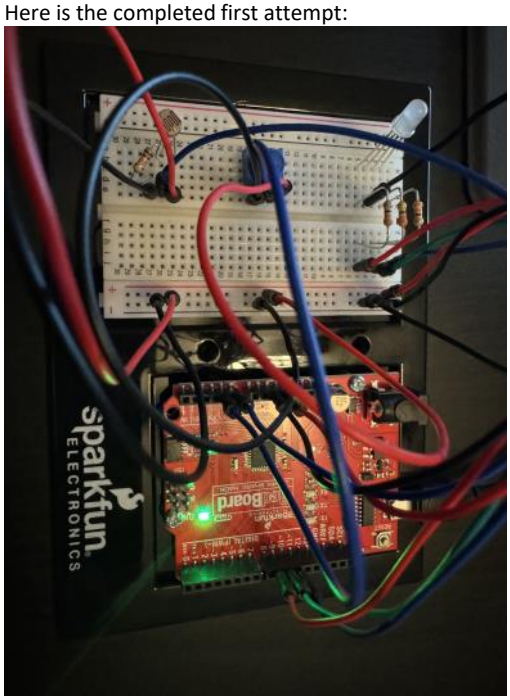
October 13, 2024 2:43 PM

A mulit-colour night light! This project introduces multi-colour LEDs (RGB LED) and pulse-width modulation (PWM).

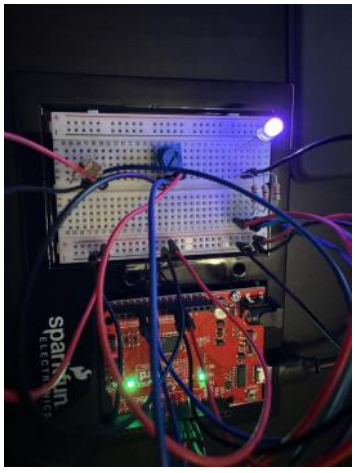
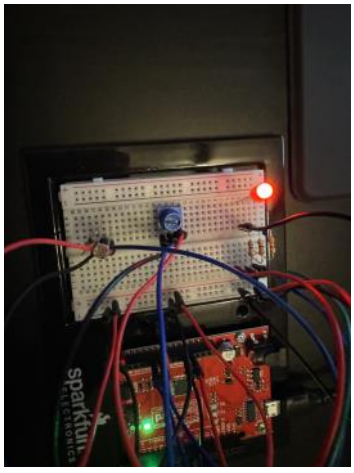
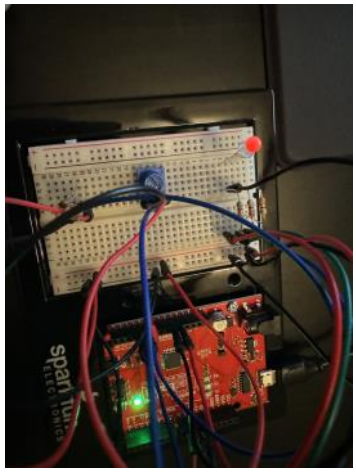
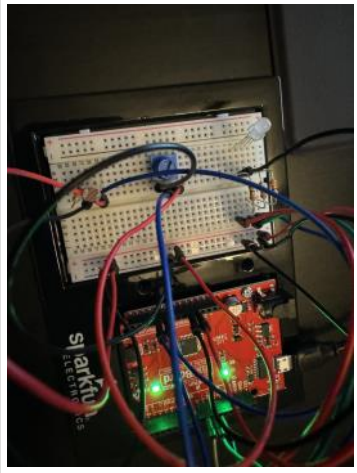
***Definition**: Pulse width modulation is a very powerful technique which allows analog variables to be controlled from a purely digital output, and with only a single data connection.*
Source: https://www.sciencedirect.com/topics/engineering/pulse-width-modulation

Note: Only a few pins on the RedBoard are capable (fast enough) for PWM, these are: 3, 5, 6, 9, 10, 11.

Attempt #1:
The setup was pretty straightforward, although we haven't had these many component on the breadboard at a time.
Here is the completed first attempt:



Unfortunately, after uploading, I found that the LED didn't operate exactly as suspected. The LED would light up, mostly. The potentiometer was able to adjust the colours of the LED, sort of. Here are the strange results of the LED:

			
Blue appears to work	Red appears to work	Partial red LED?	No LED at all!

Red and Blue appeared to light up, but the colours and LED performance didn't seem right. I disconnected from the USB and inspected the board. I found no issue with the jumpers,

potentiometer or photosensor. I knew the orientation of the potentiometer and photosensor didn't matter, but the RGB LED orientation does matter. Upon closer inspection, I had installed the RBB backwards. That's twice I've done that, I must be more careful or get a better magnifying glass!

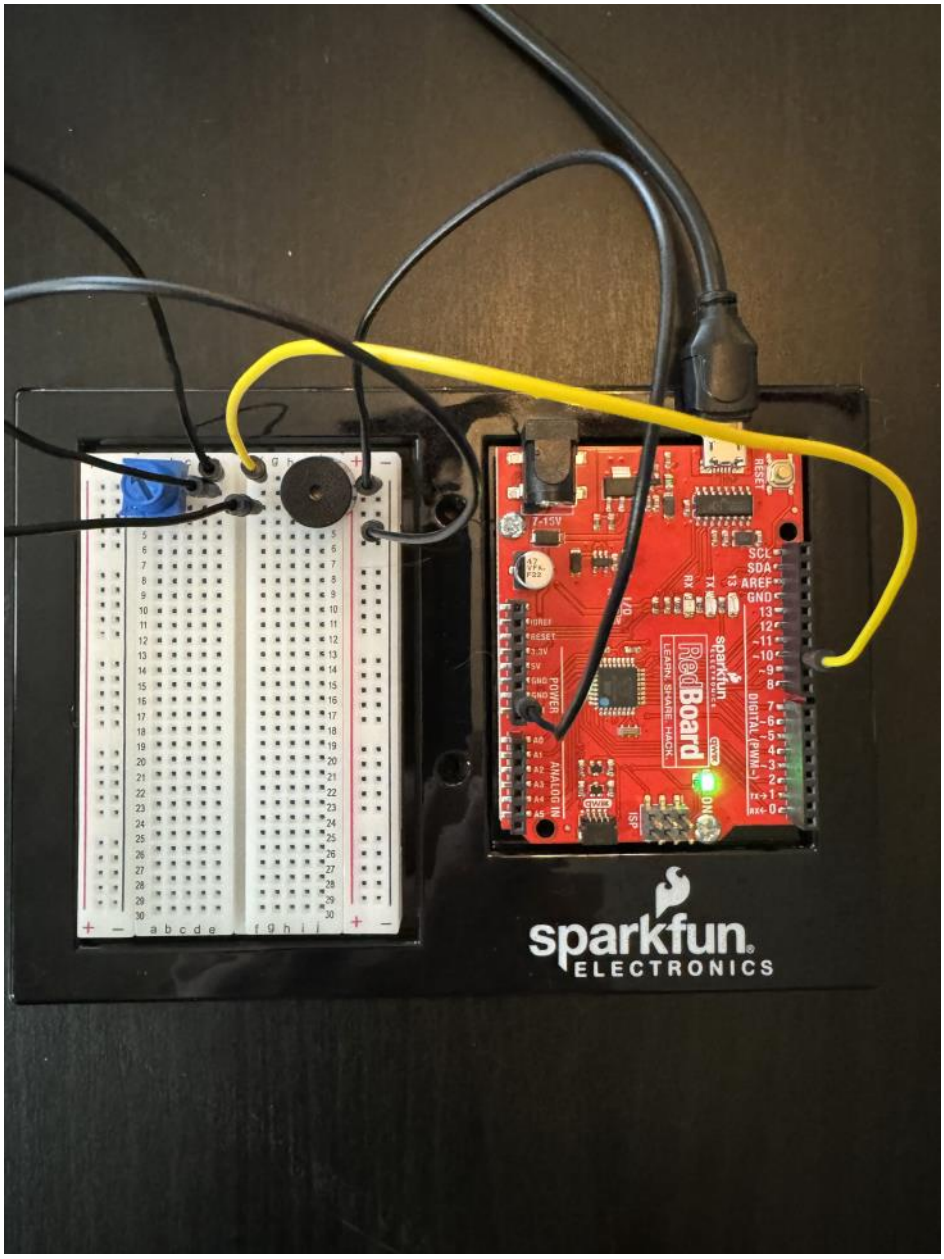
Here is the final result dialing through the colours:

Project 2B
RGB LED -...
(OneNote does not support GIF, double click to open)

Circuit 2A: Buzzer

October 13, 2024 1:47 PM

This exercise introduces the buzzer and the array coding technique. I had no problem setting up this board, and uploading the code, everything went perfect.



Coding Challenges:

Beat Length Multiplier: I found a very easy way to change the length that a beat is played. By adding a simple new variable:

```
double beatLengthMultiplier = 0.5; // multiple the lenght a tone is player for by this..
```

Then modifying the line of code in the play() function that determines the beat length:

```
  int beatLength = 150 * beatLengthMultiplier;   //the length of one beat (changing this will speed up or slow down the tempo of the song) ** HACK: Alter beat length
```

And then, the song can play a very FAST version:

```
double beatLengthMultiplier = 0.1;
Or a very SLOW version:
double beatLengthMultiplier = 5;
```

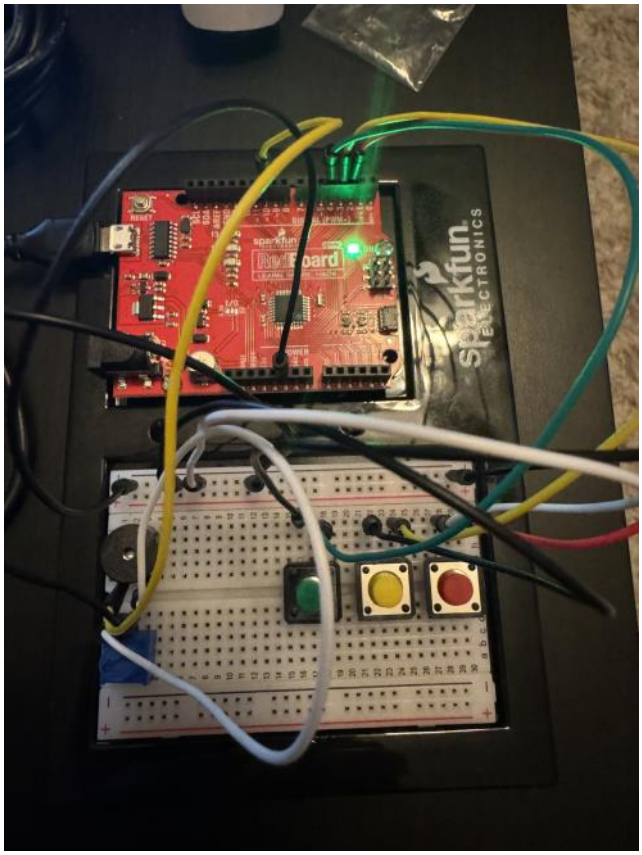
Code was tested and verified. Source code is available on git here:
https://github.com/malcolmsdad/COMP444/tree/1026481d2dadeb1114ccfb9bd9f6ef0295fe2d52/Unit%201/C2A%20Buzzer%20-%20BeatLength/SIK_Circuit_2A_Buzzer_BeatLength

Circuit 2B: Digital Trumpet

October 13, 2024 9:40 PM

The initial setup of this project was pretty straightforward, simply connected all wires as expected. This project introduces a few new concepts, including the use of buttons, pull up resisters, and binary number system.

Attempt #1: Success
The wiring and programming went smoothly, and the pietzo buzzer responded to button presses.



Coding Challenge #1: Modifying the tones. I defined a series of constant integers with the associated buzzer tone (in Hz). After uploading to the board, the program now produces different sounds for the buttons pressed.

The code is available here:
https://github.com/malcolmsdad/COMP444/tree/8f7106433b06aeed3080e606971437a1e18fdf83/Unit%201/C2B%20Trumpet%20-%20ChangeKey/SIK_Circuit_2B_DigitalTrumpet_ChangeKey

Coding Challenge #2: (Cont'd from #1):Multiple, different tones
By using several if statements, and an array I was able to analyze the buttons being collected and select a tone. The result is different tones based on the combination of buttons pressed.

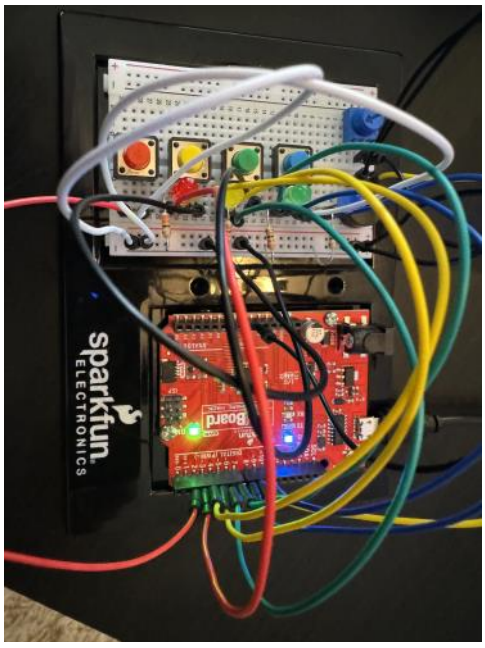
The code is available here:
https://github.com/malcolmsdad/COMP444/tree/8f7106433b06aeed3080e606971437a1e18fdf83/Unit%201/C2B%20Trumpet%20-%20MultipleTones

Circuit 2C: Simon Says

October 14, 2024 10:58 AM

A magnifying problem:
This was a very tricky one to set up. It contained 14 wires, 4 buttons and many other components. I've been finding working with the breadboard increasingly difficult for my aging eyes and oaf-like fingers, a magnifying glass helps, but often I require two hands. I found a lighted, magnifying glass visor on Amazon.ca (link here) that fit the job perfectly. I still issues navigating board when its filled with wires, but at least I can see what I'm doing!

Here is the initial board before programming:



A successful compile was a good start, the tones played perfectly, but again - for seeming the third time in a row - the LEDs didn't show. I must've reversed their polarity. That was it! Switched the LEDs polarity, and problem solved. The game works successfully; if I match 10 correct processes a winning chime is played, otherwise a losing one. I also confirmed the other behaviours such as timing out before inputs, incorrect button press (see attached GIF), the progression of correct inputs until the ultimate 10 are correct, and finally the increase in tones between sucessful rounds.

Here is an animated (no audio) GIF of a failure:

Project 2C
Simon Sa...

I'm very familiar with much of the code, although I did learn the millis() function. The helper functions are very useful as well. I suspect that creating a library of them for later use would be handy!

Coding Challenges: 1 and 2: I was easily able to increase the difficulty of the program by increasing the array length to 15. Modifying the tones was also quiet easy, however the tones I came up with don't sound particularly nice. The code is here:
https://github.com/malcolmsdad/COMP444/tree/6251fbbb95543c5cce5f61ea51cfb016e7f6a616/Unit%201/C2C%20Simon%20Says%20-%20Difficult%20Tones/SIK_Circuit_2C_SimonSays_Difficult_Tones

I'd like to highlight the following changes:

I decided to remove the int that stored the number of rounds, instead I decided to use the length of the array to determine that. So to increase difficulty of the game, simply increase the size of the array. I've increased to 20 rounds here:

```
// This variable isn't needed! We could analyze the length to track number of
required wins!
//int roundsToWin = 10;         //number of rounds the player has to play
before they win the game (the array can only hold up to 16 rounds)
// the length of the button sequence will now determined the number of rounds
to play
int buttonSequence[20];       //make an array of numbers that will be the
sequence that the player needs to remember
```

I also created a helper function to easily determine the array length (instead of including STD):

```
// calculate the length of the array based on the size of the array allocated
divided by the size of an element.
int gameSize(){
  return sizeof(buttonSequence)/sizeof(buttonSequence[0]);
}
```

And finally, I made an absolutely horrible random winning music function, it basically plays a random number of tones, a random number of times:

```
// winMusicRandom: This function will play a random tone a random number of
times. Will sound very strange.
void winMusicRandom()
{
  for (int i=0; i<random(0, 8); i++){           // play a random number of
tones
    tone(buzzerPin, random(1000, 3100), 150);   //play a random tone
    delay(175);
  }
}
```