Spaceship Interface

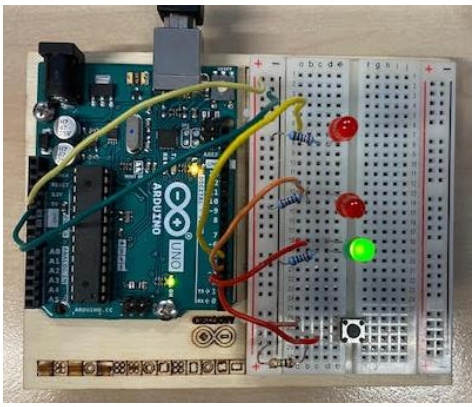

Figure 1) Setup of the Spaceship Interface when the switch is open

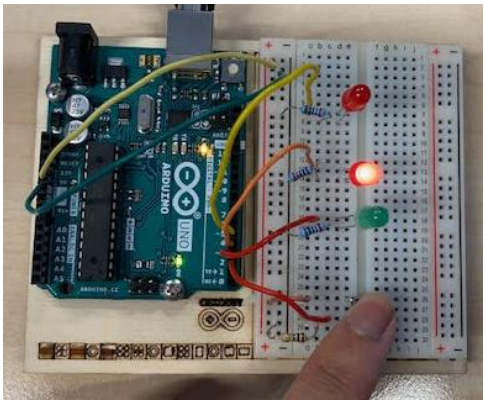

Figure 2) Button pressed which activates the red LEDs

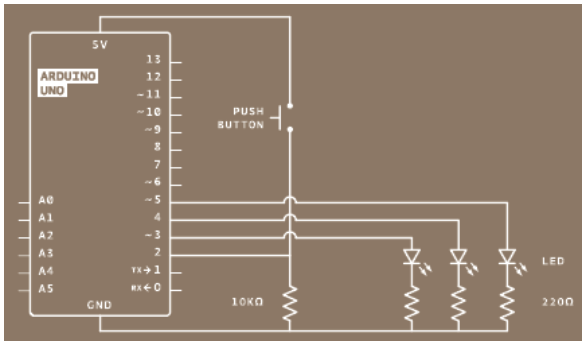

Figure 3) Circuit diagram of Spaceship Interface

Connection of the Board

A wire is connected from the 5V pin on the Arduino to the positive part of the long bus lines in the breadboard and a wire is connected from the ground pin to the negative part of the long bus lines. Two red LEDs and one green LED are placed on the breadboard. For each LED, the cathode is connected to ground through a 220-ohm resistor and the anodes are connected to pin 3,4 and 5 where pin 3 is green and the other two are red. The switch is placed on the breadboard where one side is connected to power and the other is to digital pin 2, as well as, a 10-kilohm resistor is connected to ground and to the switch pin that connects to the Arduino. This is done so that when the switch is open, it reads LOW when there is no voltage coming in through the switch.

Function of the Project

The project is meant to imitate a 1970s science fiction spaceship interface. The program begins by naming the button on the breadboard, switchState, and setting it to 0. All three LEDs are set to OUTPUT and the button is set to INPUT in the setup. At the beginning of the loop, the program reads the value of the button and assigns it to switchState. A IF statement then determines the actions the program will take. If switchState reads LOW, only the green LED will be turned on. Otherwise, the green LED will be turned off the red LEDs will blink, meaning one LED will be on at one time with a quarter second between one of the LEDs turning off and the other turning on.

Other possible applications

The system could be integrated into a crosswalk system. This can be done replacing one of the red LEDs with a yellow one. However, when the circuit is open, the red LED will be on first. Also, instead of holding down the button you press it once. Once it's been pressed the yellow LED can turn on for a few seconds to signal that you need to wait and that you've pressed the button. A timer will countdown from when you pressed the button and as it hits zero, the red LED turns off and the green LED turns on to signal that you can cross the road.

Love-o-Meter

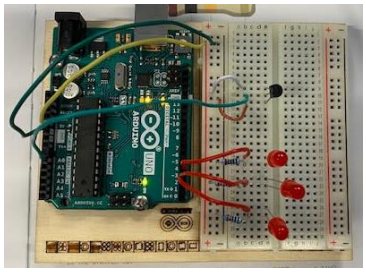

Figure 4) Setup of Love-o-Meter

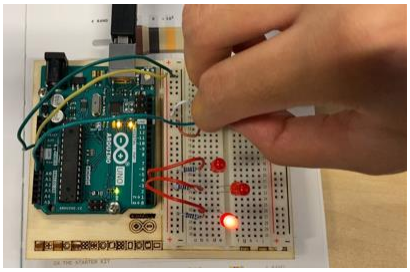

Figure 5) Love-o-meter with one LED is on

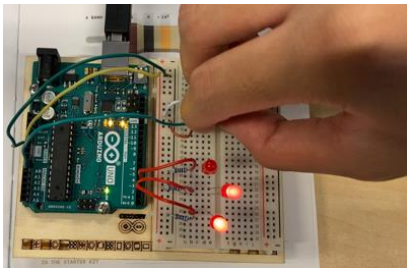

Figure 6) Love-o-meter with two LEDs on

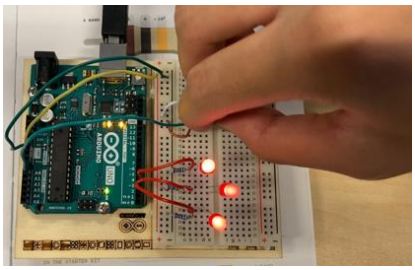

Figure 7) Love-o-meter with three LEDs on

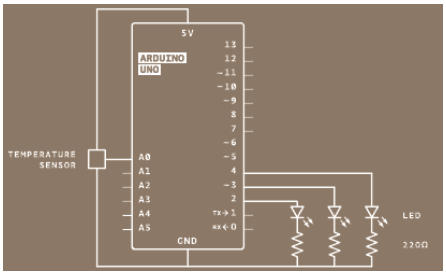

Figure 8) Circuit diagram of Love-o-Meter

Connection of the Board

Like in the Spaceship Interface project, the breadboard is wired up to power (5V) and ground. We then use three red LEDs and connect each cathode to ground through a 220-ohm resistor. Each anode of the LEDs is connected to pins 2 through 4. Next, the TMP36 is placed on the breadboard with the rounded part facing away from the Arduino board. The left pin is connected to power, the right pin to ground and the centre pin to pin A0 on the Arduino.

Function of the Project

The project is about using a temperature sensor to light up three red LEDs to determine how "hot" you are. The variable sensorPin is set to pin A0 on the Arduino board. sensorPin will be used to the read the value of the temperature sensor. A variable baselineTemp equals 28, which will be used to calibrate the red LEDs. In the setup, the serial port is opened, and a FOR loop is used to set some pins as OUTPUTs. These pins are attached to the LEDs and are assigned variable names as numbers by iterating through 2 to 4. Each pin is then set to LOW. The loop section starts by getting the value of the sensorPin, storing it in sensorVal and displays it. A variable called voltage is created where sensorVal is used in an equation to calculate voltage and is then displayed. This process is repeated when creating the variable temperature, but the variable voltage is used to calculate it and is then displayed as well. After this is done, an IF statement followed by three ELSE IF functions is used. Depending on the temperature reading, one, two or three LEDs will light up.

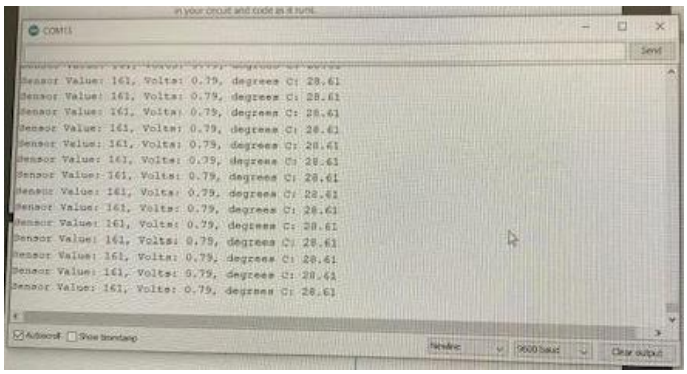

Figure 9) Image of Monitor display showing the Sensor Value, Voltage and Temperature

Other possible applications

Can be used to monitor the temperature of a computer. However, you wouldn't need the LEDs. This would be helpful for the user as it would tell them if the cooling within the PC is substantial enough to stop it from overheating.

Colour Mixing Lamp

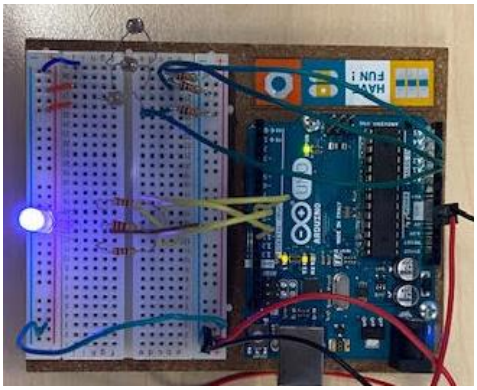

Figure 10) Setup of the Colour Mixing Lamp with the Arduino board plugged in with the code uploaded too

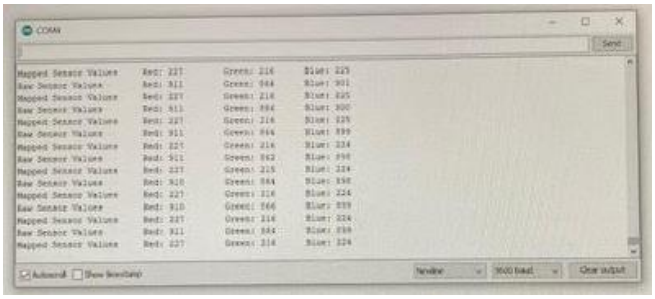

Figure 11) Serial Monitor displaying the RGB values

Connection of the Board

5V and ground are attached to one side of the breadboard from the Arduino. Three photoresistors are placed on the breadboard where they cross the centre divide from one side to the other, as shown in Figure 10. One end of each photoresistor is attached to power. The side is attached to a 10-kilohm resistor to ground. The resistors are in series with the photoresistor, and together they form a voltage divider. The resistance of the photoresistor changes when light hits and the voltage at this junction changes as well. On the same side as the resistor, the photoresistors are connected to Analog pins 0 through 2 with the hookup wire. Three coloured gels are placed one over each of the photoresistors. The red gel is placed over the photoresistor that is connect to A0, the green over the one connected to A1 and the blue over the one connected to A2. These filters only let light of a specific wavelength through to the sensor its covering. The red filter only allows red light, green only allows green light and blue only allows blue light. In the LED, you can create a voltage difference between the cathode and the voltage coming out of the Arduino's PWM pins, which will cause the LED to fade between its three colours. The LED is connected to the board in the following way. The longest pin of the LED is placed on the breadboard and is connected to ground. The other three pins are connected to digital pins 9,10, and 11 in series with 220-ohm resistors.

Function of the Project

In the colour mixing lamp project, a tri-colour LED and three photoresistors are used to create a lamp that changes colour depending on external lighting conditions. The code starts by creating 6 variables. 3 for each pin on the LED and 3 for the sensor pins according to their colour and sets it to a pin on the Arduino board. Another 6 variables are made which will be used to store the readings of the LED and photoresistors. In the setup, each LED pin is set to OUTPUT. Secondly, in the loop, the light sensor readings are read and displayed. Arithmetic is used on the sensor readings and are stored in new variables. These variables are displayed and are used to change the colour of the LED.

Other possible applications

Can be used to identify colours of a fabric for a machine that sorts out things into colours.

Mood Cue

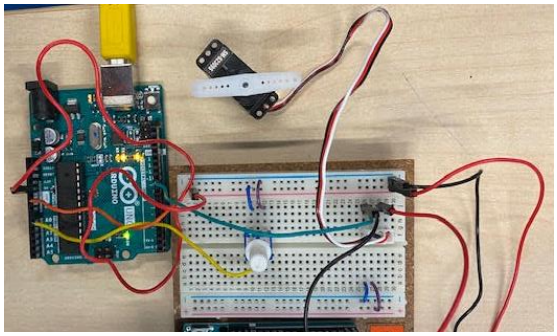

Figure 12) Setup of the Mood Cue project

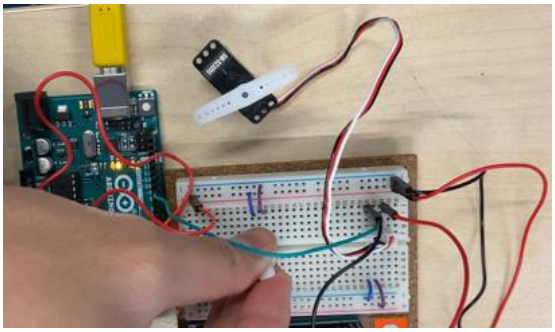

Figure 13) Turning the potentiometer which rotates the servo

Figure 14) Image of serial monitor displaying the values of potVal and angle

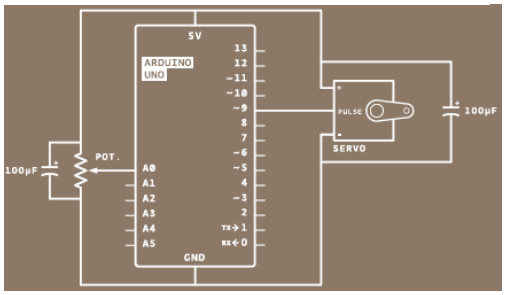

Figure 15) Circuit diagram of Mood cue

Connection of the Board

5V and ground are attached to one side of the breadboard from the Arduino. A potentiometer is placed on the breadboard. One side is connected to power and the other to ground. The middle pin is connected to pin A0, which will control the position of the servo. The servo is connected to the board through three male headers that are plugged into the female ends of the servo wires. The headers are connected to the breadboard where each pin is in a different row. 5V is connected to the red wire, ground to the black and white to pin 9.

Function of the Project

The Mood Cue project makes use of a servo motor to make a mechanical gauge point out what sort of mood you're in that day. In the Arduino code, the Servo library is imported into Arduino. Three variables are created and are set as integers. The variables are potPin, which equals A0, potVal and angle. In the setup, the servo is set pin 9 and the serial port is opened. In the loop, the value of potPin is retrieved and stored in potVal which is then displayed. To create a usable value for the servo from the analog input, the map function is used. This function scales numbers. The function takes in five arguments which are potVal, 0, 1023, 0, 179 in the form **map(potVal, 0, 1023, 0, 179)**. This will convert values between 0-1023 to 0-179. The value created is stored in the angle variable and is displayed. To move the servo, the command **myServo.write()** is used with the variable angle used to specify how much it should. A delay is inserted to allow the servo to move to its new position.

Other possible applications

Could be converted into a spin the wheel game, however you would replace the potentiometer with a button. This button will cause the servo to spin for a random duration and stop.

Light Theremin

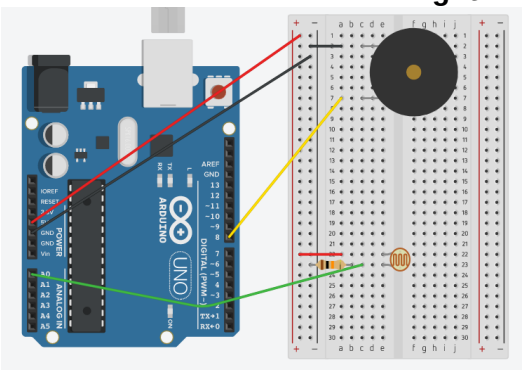

Figure 16) Setup of Light Theremin project

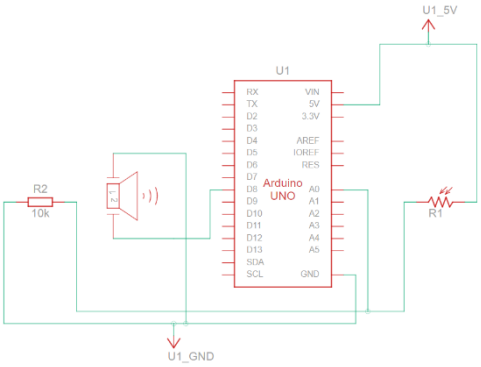

Figure 17) Circuit diagram of Light Theremin project

Connection of the board

The outer bus lines are connected to power and ground. With the piezo, one end is connected to ground and the other is connected to digital pin 8 on the Arduino. The photoresistor is placed on the breadboard, with one end connected to 5V. The other end is connected to analog pin 0 and to ground through a 10-kilohm resistor.

Function of the project

A photoresistor and a piezo element is used to make light-based theremin – a theremin is an instrument that makes sounds based on the movements of a musician's hands around the instrument. In the Arduino code, a variable holding the analogRead() value from the resistor and another called sensorHigh storing a value 0 is set to calibrate the sensor. A constant is created named ledPin which will be used as a calibration indicator. In the setup(), the pinMode() is changed of the ledPin to OUTPUT and the light is turned on. A while() loop is then ran to loop for 5 seconds until a condition is met for calibration. millis() function is used in the loop to check the current time. Then the sensor values are compared for calibration as well. In the loop, the sensor value is read and if the value is less than sensorLow, the variable will be updated. But, if its greater than sensorHigh, that gets updated. We'll then indicate calibration has finished. The program will wait 5 seconds and the while() loop will end. The LED attached to pin 13 will turn off and the sensor high and low values are recorded to scale the frequency. In the loop(), the value on A0 is read and stored in the sensorValue. The sensor value is then mapped to a frequency. A variable named pitch is mapped from sensorValue. sensorLow and sensorHigh are used as the bounds for the incoming values. The starting values for the output are set from 50 to 4000. This will set the range of frequencies. Next, the frequency is played. The function tone() is called to play a sound. It takes three arguments: what pin to play the sound on, what frequency to play and how long to play the note. A delay() is then called for 10 milliseconds to give the sound some time to play.

Other possible applications

The project can be altered to mimic the sound of a doorbell. You do this by removing the photoresistor and using a button instead. In the program you can alter the sound of the piezo to play a sequence of sounds to your liking.

Keyboard Instrument

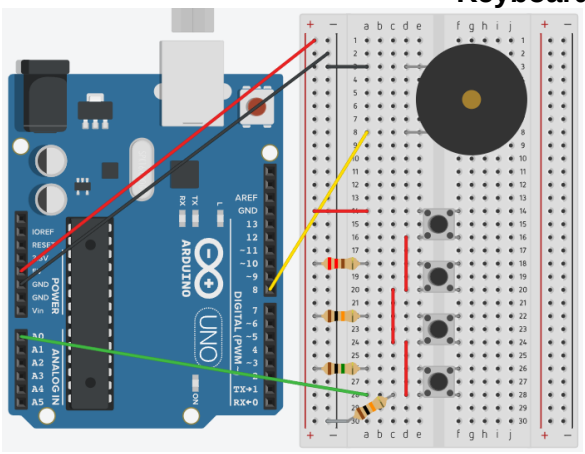

Figure 18) Setup of Keyboard Instrument project

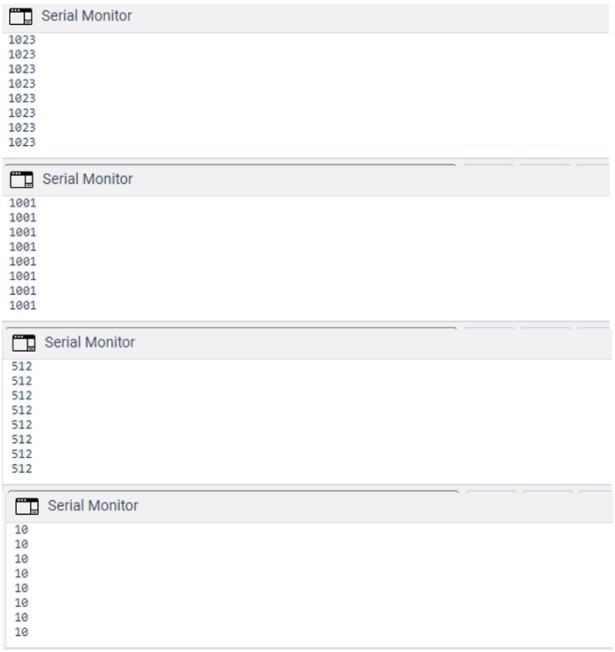

Figure 19) Screenshot of serial monitor when each switch on the breadboard is pressed down

Connection of the board

The breadboard is wired up to power and ground. One end of the piezo is connected to ground and the other to pin 8 on the Arduino. The switches are then placed on the board as shown in Figure 16. The resistors and switches are arranged in a resistor ladder where they feed into an analog input. The first switch is connected directly to power. The second, third and fourth switches are connected to power through a 220-ohm, 10-kilohm and 1-megohm resistor, respectively. The outputs of the switches are connected in one junction, which is connected to ground with a 10-kilohm resistor and to analog pin 0.

Function of the project

Four resistors and buttons are used to build a small keyboard which emits sound by using a piezo. In the Arduino code, an array of frequencies is created. The array contains the frequencies: 262Hz, 294Hz, 330Hz and 349Hz. The array is made to be a global variable and is declared before setup(). In setup(), the serial communication starts. In the loop(), the analog value is read and sent to the serial monito using the function Serial.println(keyVal). An if() and else statement is used to determine what note is being played. This is done by assigning each value a different tone. The program is then configured to play the notes that correspond to the analog value by using if() statements as well. After each statement, the tone() function is called. The program references the array to determine what frequency to play. If the value of A0 matches one of the if statements, the program will play the tone. To stop playing the tone when nothing is pressed we call the function noTone() as well as providing the pin number of the note that was being played.

Other possible applications

No other possible applications have been thought for this Arduino system as it was very specifically to replicate a piano. So to make any proper changes, would change the system completely as a whole.

Digital Hourglass

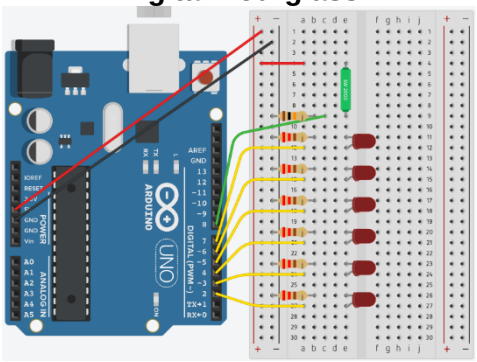

Figure 20) Setup of Digital Hourglass

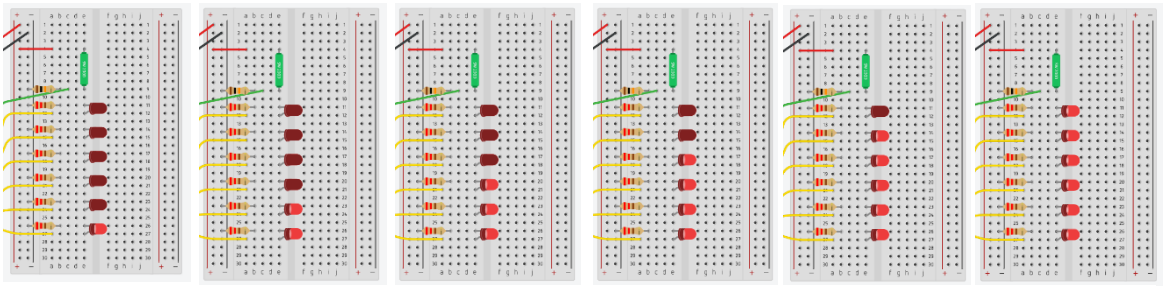

Figure 21) Screenshots of the simulation of the project using TinkerCAD. Each LED represents the amount of time that has passed but also how much time is left as well.

Connection of the board

The breadboard is connected to power and ground. The anode of six LEDs is connected to digital pins 2-7 and are connected to ground through 220-ohm resistors.

Function of the project

Six LEDs and a tilt sensor are used to create a digital hourglass, that turns on an LED every ten minutes. In the Arduino code, a constant named switchPin is created. Another variable of type unsigned long is created to hold the time an LED was last changed. Variables for switch state and previous switch state are created which be used to compare the switches position from one loop to the next. Another variable named led is created, which will be used to count which LED is the next one to be turned on. The last variable declared is the interval between each LED turning on. This will be of the long datatype. In the setup() the LED pins are declared as OUTPUTs and the switchPin as an INPUT. In the loop(), the time is checked since the program started using the function millis() and is stored in a local variable named currentTime. An if() statement is used to evaluate the amount of time that passed since the previous loop(). This is done by subtracting the currentTime from the previousTime and checking if its greater than the interval variable. Next, you turn on an LED and prepare for the next one. Eventually, the program checks to see if all the lights are on. An if() statement is used to check if the switch is in a different state that the position it was previously in. If they are different, the LEDs are turned off and the led variable is returned to the first pin and the timer is reset for the LEDs. At the end of the loop() the current state is set to the previous state.

Other possible applications

You could turn digital hourglass system into a mini-racing light system. You change one of LEDs to green and instead of the program counting to one hour, it can count to 6-seconds for example and each light will turn on and the last one to turn on will be green. This light system can be used for Scalextric track to indicate to the players of the game when to go.

Motorised Pinwheel

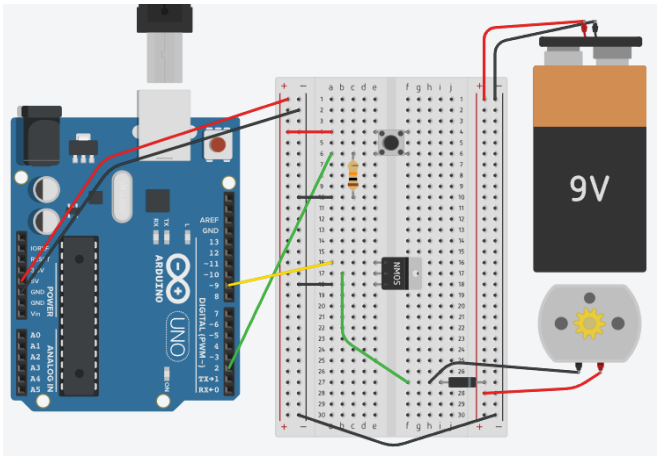

Figure 22) Setup of Motorised Pinwheel Project

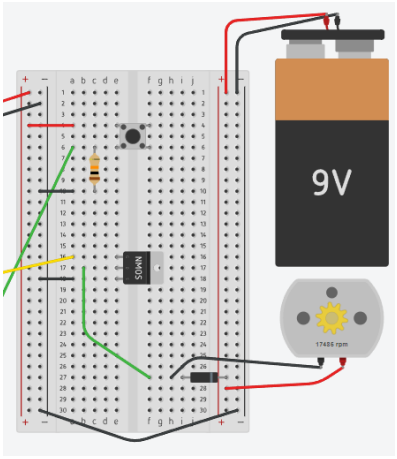

Figure 23) Screenshot of simulation when the button is pressed, causing the motor to rotate.

Connection of the board

The power and ground are connected to the breadboard through the Arduino. A switch is added to the board and one side is connected to power and the other to digital pin 2. A 10-kilohm pull-down resistor is also connected to ground on the output pin of the switch. We then plug a 9V battery snap into the breadboard and connect ground from the battery to the ground of the Arduino on the breadboard with a jumper. The motor's free lead is then attached to the 9V power. The transistor is placed on the board with the metal tab facing away from you. The left pin is connected to digital pin 9. This pin is called the gate. One end of the motor is connected to the middle pin of the transistor. This pin is called the drain. The last pin is connected to ground and is called the source. The motor's voltage supply is connected to the motor and breadboard. Lastly, the diode is added to the breadboard. The anode is connected to the ground of the motor and the cathode to the power of the motor.

Function of the project

In this project, a motor is used to spin a colourful pinwheel. In the Arduino code, the constants for the switch and motor are setup, as well as a variable named switchState is created. In the setup(), the pinMode() of the motor is declared as an OUTPUT and the switch as INPUT. In the loop(), the state of the switchPIN is checked with digitalRead(). Using an if() statement we determine the state of the motor. If the switch is pressed, the motorPin is set to HIGH and if it's not pressed it will be turned to LOW. Also, when HIGH, the transistor will activate, completing the motor circuit and when LOW the motor will not spin.

Other possible applications

This system can be used as a mini fan. All you must do is attach a fan at the end of the motor and you have completed it.

Zoetrope

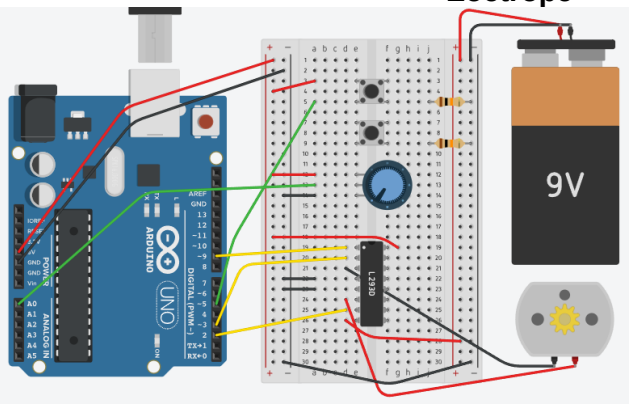

Figure 24) Setup of the Zoetrope project

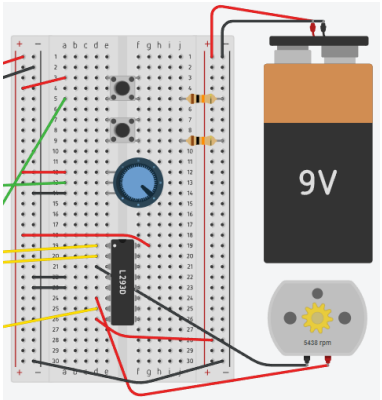

Figure 25) Screenshot of project working using the TinkerCAD simulation

Connection of the board

Power and ground are connected from the breadboard to the Arduino. Two switches are then added with one side of the switch connected to power. A 10-kilohm resistor is connected in series with ground on the output pin of both switches. The switch on pin 4 will control the direction and the switch on pin 5 will turn the motor on and off. Next, the potentiometer is connected to the breadboard with one side connected to power and the other to ground. The centre pin is connected to analog input 0, which be used to control the speed of the motor. The H-bridge is placed in the centre of the breadboard as shown in Figure 22. Pin 1 of the H-bridge is connected to digital pin 9. Pin 2 is connected to digital pin 3, pin 7 to digital pin 2. These two controls the direction the motor spins in. Pin 16 is connected into power and pins 4 and 5 to ground. The motor is attached to pins 3 and 6 on the H-bridge as well. Lastly, the battery connector is plugged to the other power rails on the breadboard. Ground from the Arduino is connected to the battery's ground and pin 8 from the H-bridge is connected to the battery power.

Function of the project

In this project moving images are created using a motor connected to a H-bridge with some still images. In the Arduino code, we start by creating the constants for the output and input pins. Variables are then created for remembering the program state. Next, variables for motor control are created. The following names motorDirection and motorPower are used. Then, in setup(), the digital pins are declared as inputs and outputs. The motor is turned off so that the motor isn't spinning right away. This is done by turning the enable pin LOW to start. In the loop(), the sensor information is read and stored in the onOffSwitchState variable. Next, the program checks whether the on/off sensor has changed. At this stage, motorPower will be set to 1 or 0 if the reading is HIGH or LOW respectively. Then, the program checks to see if the direction has changed. Depending on the motorDirection variable, the direction of the motor will continue to rotate in the same way or it will be in reverse. Next, PWM the motor if it is enabled. This is controlled using analogWrite() and the variable motorEnabled and changing the value to 1 or 0 to turn the motor on or off respectively. Lastly, before exiting the loop() the current state of the switches is saved for the next run through of the program.

Other possible applications

You could change this into a remote control for a mini car. You'll just have to do additional coding and add a few more components to the Arduino system to allow the device to communicate with the motor remotely. The buttons can continue to have the same function; however, the activation of the reverse-button should be set to hold instead of press. This will allow the user to have better control of the motor. To stop reversing, you must let go of the button. The function of the potentiometer can stay, and it can be used to drive the car. The more you rotate it the faster it goes and if you turn it back the other way it slows down and can stop if rotate it completely.

Crystal Ball

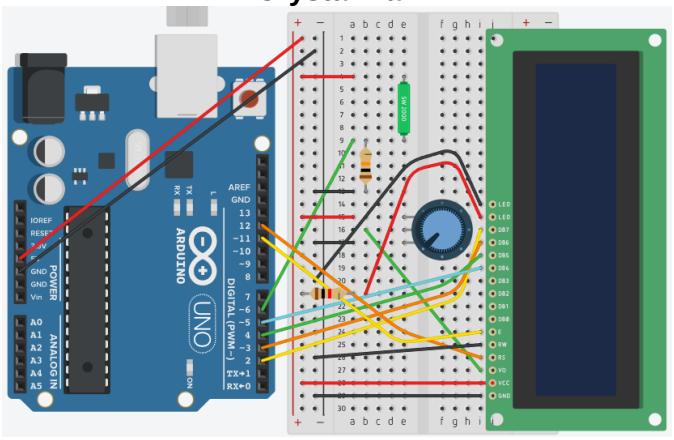

Figure 26) Setup of Crystal Ball project

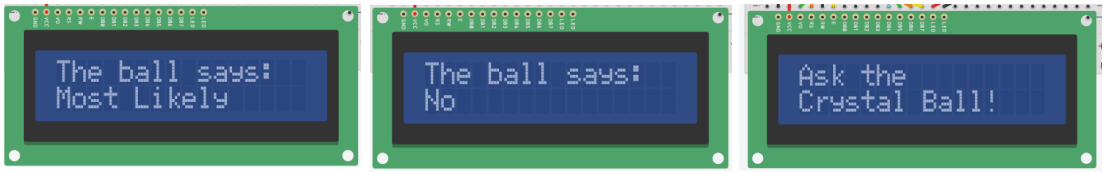

Figure 27) Screenshots of the LCD screen during the simulation of the project. Each screen shows some of the text that can appear while using the device.

Connection of the board

The power and ground are connected to one side of the board. A tilt switch is placed on the breadboard and one side is attached to 5V and the other to ground through a 10-kilohm resistor and digital pin 6 on the Arduino. For the LCD component, the following pins will be used: register select, read/write, enable and data pins D4-D7 and the potentiometer will be used to adjust the contrast. The two outside pins of the LCD, Vss and LED- are connected to ground. The R/W pin is connected to ground, Vcc to 5V directly and the LED+ pin is connected to power through a 220-ohm resistor. The Arduino digitals pins, 2-5 are connected to LCD D7-D4 respectively. These data pins will tell the screen what character to display. Then EN on the screen is connected to pin 11 on the Arduino and RS to pin 12. Lastly, the potentiometer is placed on the breadboard with one end connected to power and the other to ground. The centre pin connected to V0 on the LCD. This will allow you to change the contrast of the screen.

Function of the project

In this project and LCD screen and Tilt Sensor is used to create a crystal ball. The LCD is used to display the random messages and the sensor is mimic the shaking of the crystal ball. In the Arduino code, you start by importing the LiquidCrystal library which is responsible for simplifying the process of writing software to display characters. In setup(), we declare the switch pin as an INPUT. Then, the print() function is used to display introductory screen. However, the cursor must be moved. To position the text correctly on the screen. The setCursor() function is used to do this. When the program starts, the loop() starts by checking the switch first and the value is put in switchState. Then, a random answer is chosen by using an if() statement and the function random(). Depending on the switch position the screen will display a new message or not. To display the message, switch() statements are used to execute different pieces of code. These different pieces of code is called a case.switch(). Depending on the variable, a certain case statement will be executed. In the case statements, lcd.print() is used with addition of the break command. This allows the program to skip to the end of the switch statement. Lastly, the switch state is assigned to prevSwitchState to carry into the next loop.

Other possible applications

No other possible applications have been thought for this Arduino system

Created Project – Tilt Lamp

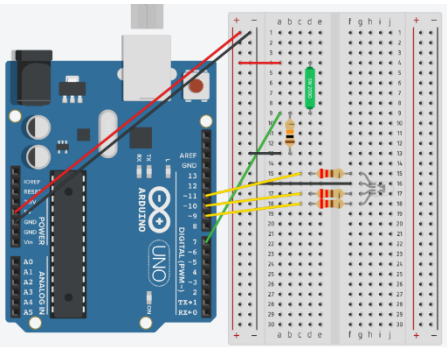

Figure 28) Setup of created project called Tilt Lamp

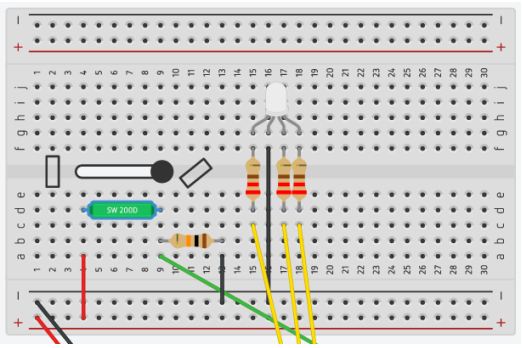

Figure 29) Screenshot of the device working. When the sensor detects a tilt, the LED will turn on

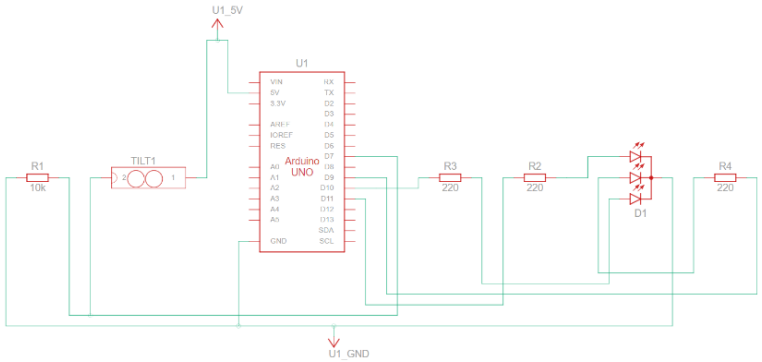

Figure 30) Circuit diagram of Tilt Lamp

Connection of the board

The power and ground are connected to one side of the board. A tilt sensor is placed on the bread board with one end connected to power and the other to ground through a 10-kilohm resistor. Also, it is connected to digital pin 7. An RBG LED is placed on the board with the red, blue and green pins connected to digital pins 9, 10 and 11 through 220-ohm resistors each, respectively.

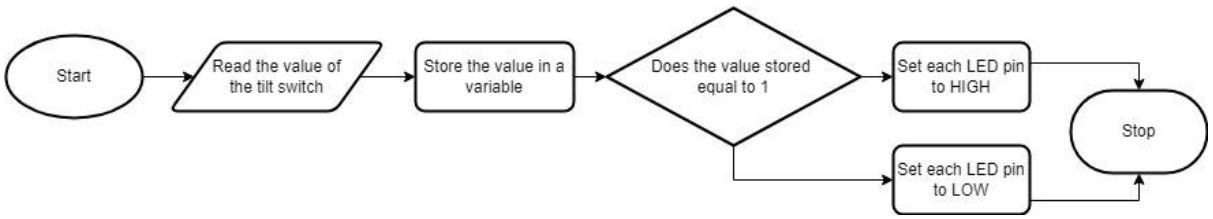

Figure 31) Program logic diagram of Tilt Lamp project

Function of the project

In this project a tilt sensor and a RGB LED are used to make a lamp that turns on by turning it the other way around. In the Arduino code, constants for each pin on the RGB LED and tilt switch are created. These will be used to control the components that are placed on the breadboard. To read the state of the tilt switch a variable called switchState is created to store this value. In the setup(), all three LED pins are declared as OUTPUTs and the variable switchPin is declared as an INPUT. In the loop(), the program starts by reading switchPin and storing the value in switchState. An if()… else statement is created to decide whether to turn on the LED or not. If the switchState value reads 1, all the LEDs will be set to HIGH, otherwise set it to LOW.

Case Study of a Mechatronic System - Building your own quadrotor [1]

Function of the system

This system aims to create quadcopter from scratch rather than using a commercial made one. A quadcopter is an aerial vehicle that has four propellers, which are positioned horizontal to the ground. The system combines the use of the knowledge of flight mechanics, brushless motors, propellers, an electronic speed controller, an IMU sensor, battery and control board to make this device work.

To understand force distribution on the system as well as how to move it, Figures 32 and 33 present this.

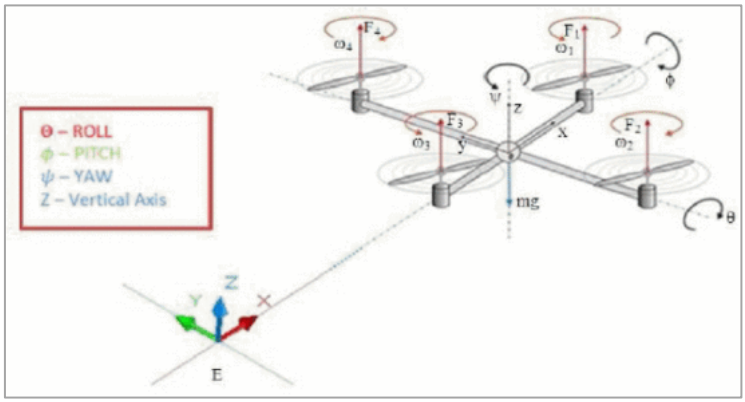

Figure 32) Free-Body forces [2]

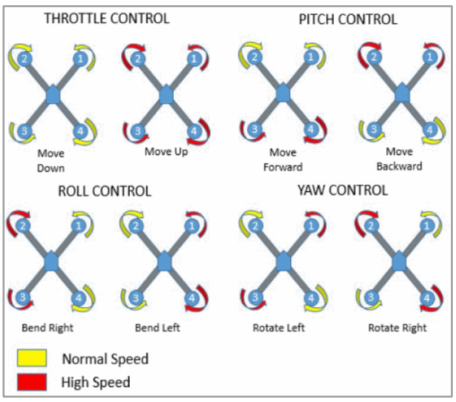

Figure 33) Basic movements of a quadrotor

Figure 32 shows the different forces acting on the system which are thrust forces and weight. Also, the basic motions of the quadrotor, as shown in Figure 33, are obtained by doing a summation of forces based on the forces shown in Figure 32 [3][1].

To calculate the right values, we need to use to ensure the system works, the following equations are used:

$$F1+F2+F3+F4=W=m\times g=FT \quad (1)$$

$$F1+F2+F3+F4>W \quad (2)$$

$$F1+F2+F3+F4<W \quad (3)$$

Equation (1) describes the total thrust force that must be generated by the propeller for the device to hover. In Equations (2) and (3), they describe the climb and descend movements [3]. You also must calculate the values that determine the motor speed of the quadrotor depending on the direction you want to move in. The aircraft can move on three axes: pitch, roll and yaw.

Roll:

$$W1=FT-offset$$
$$W2=FT-offset$$
$$W3=FT+offset$$
$$W4=FT+offset \quad (4)$$

Pitch:

$$W1=FT-offset$$
$$W2=FT+offset$$
$$W3=FT+offset$$
$$W4=FT-offset \quad (5)$$

Yaw:

$$W1=FT-offset$$
$$W2=FT+offset$$
$$W3=FT-offset$$
$$W4=FT+offset \quad (6)$$

where:
W = Motor speed
FT = Throttle setting
[1]

You also need to know the equations of motion that describe the roll and pitch dynamics of the quadrotor. This is done by applying the Newton-Euler formula to the system of Figure 32.

$$\Phi^{\cdot\cdot}=(Iyy Izz/Ixx)\theta\psi-(Jtp/Ixx)\theta\Omega+l(U2/Ixx) \quad (8)$$
$$\theta^{\cdot\cdot}=(Ixx Izz/Iyy)\theta\psi-(Jtp/Iyy)\theta\Omega+l(U3/Iyy) \quad (9)$$
$$U2=f2-f4$$
$$U3=f1-f3 \quad (10)$$

Where: fi = thrust, l = arm length, Ixx = Rotational inertia along x axis, Iyy = Rotational inertia along y axis, Izz = Rotational inertia along z axis.

Modules of the system

For the design of the aircraft, a commercial carbon fibre frame called “bumblebee” was used [1]. In this section, it will include further detail of modules of system of the Quadrotor.

Figure 34) Real life model of the Quadcopter

Actuators:
The bumblebee frame includes four brushless motors, Table 1 shows the motor specifications. From the table, tt can be concluded that these motors can generate the thrust force necessary to make the quadrotor hover, knowing that the total weight of the aircraft is 1400 grams.
Propellers:
Due to the specifications of the motor, a set of 10 X 3.8 inch propellers was chosen.
Electronic Speed Controller:
An ESC that can handle 40 amperes is used to support the required current to produce 1000 grams of thrust per motor.
Sensors:
The aircraft contains an onboard IMU (inertial measurement unit) that has 9 degrees of freedom, integrating accelerometer, gyro and compass functions. (Table 2 contains the IMU specifications)
Control Board:
The microcontroller used to control the aircraft was Arduino Uno. (Table 3 contains the Arduino specifications)
Software:
The serial communication the quadcopter will be using is Xbee, which will enable acquisition and monitoring of angle data using Matlab.
Control Design:
A PID controller is used as the control strategy, which is based upon equations that can modelled and used to compare to other data.

References

[1] G. H. Tuta Navajas and S. Roa Prada, "Building your own quadrotor: A mechatronics system design case study," 2014 III International Congress of Engineering Mechatronics and Automation (CIIMA), 2014, pp. 1-5, doi: 10.1109/CIIMA.2014.6983444.

[2] M. Hassan, S. Faiz, D. Hazry, Faizan A. Warsi and M. Kamran Joyo, "Stabilized controller design for attitude and altitude controlling of quad-rotor under disturbance and noisy conditions", American Journal of Applied Sciences, vol. 10, no. 8, pp. 819-831, 2013

[3] R. Buchi, "Fascination quadrocopter", Basics-Electronics-Flight Experience, 2011