PHYS 319 Project Progress Report

Dawn Zheng March 26, 2024

Objective:

* Art installation of eyes drawn on a projector that follow a person around as they move from left to right

A grid of black eyes

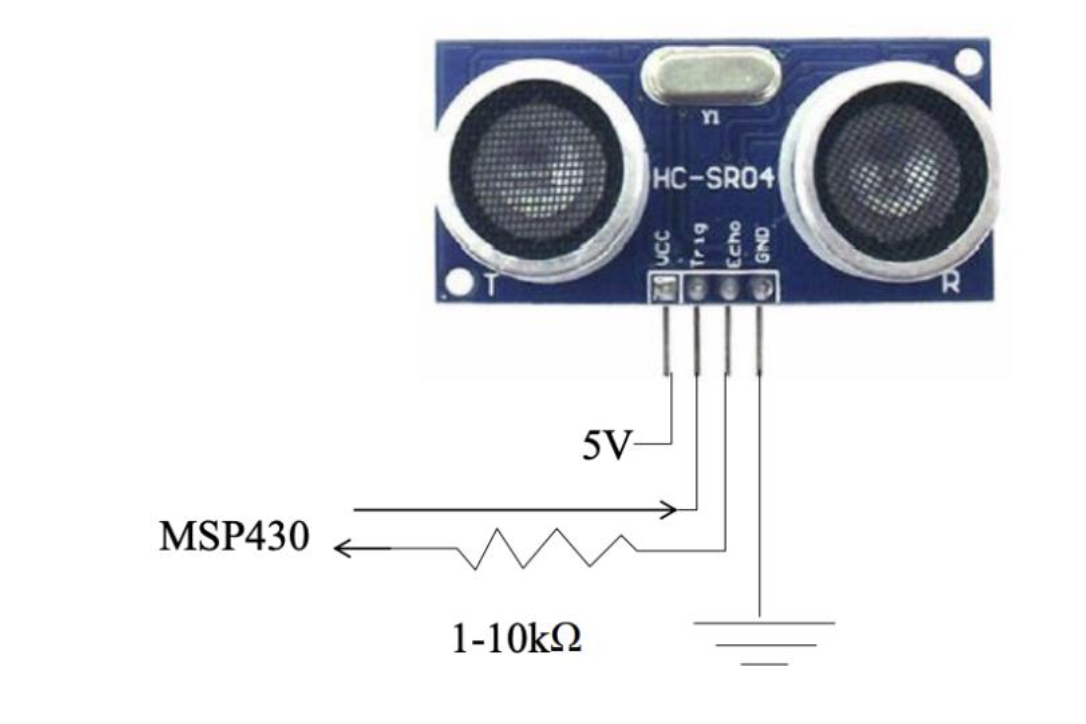
Description automatically generated with medium confidence

Plan:

* Build a circuit with an ultrasonic distance detector connected to the MSP430, set it x meters away from the projector
* As the distance measurements are transferred from the MSP430 to the computer, a python program using the numpy and pygame modules

Circuit:

* Read the SR04 as well as MSP430 documentations for timing, how to build the circuit, etc
* <https://www.youtube.com/watch?v=LzOVfDQaol8&t=622s>
* <https://www.youtube.com/watch?v=uxoA6WahESo>
* The SR04 is shown below with the appropriate connections needed



* A circuit board with wires connected to it

  Description automatically generatedThe echo SR04 is connected to the MSP430 via P6.0, with a resistor connecting the two. The Trig pin of the SR04 is connected to the MSP430 through P1.2. An image of the circuit is shown below:
* The program aims to trigger a measurement from the SR04 with a 10 us long pulse. The sensor then raises the echo pin on the SR04, which is the trigger output. The echo sends the distance measurement to the laptop. The program then waits 10 ms before repeating another measurement. The measurement is divided by 58 to convert the units to centimeters.
* Currently, the program is experiencing some timing errors which will be fixed via further reading the documentation. The non-working program is shown:

#include <msp430.h>

// the microcontroller (ie your program) triggers a measurement with a ~ 10

μs long pulse on the trigger pin.

// the sensor will initiate a measurement and then raise the echo pin

// when an echo returns (or the sensor gets tired of waiting, ~36 ms), the

sensor lowers the echo pin.

// wait 10 ms or more before repeating.

/\*\*

\* main.c

\*/#define TXD BIT4 // TXD on P4.4

(transmits data to computer)

unsigned int TXByte;

int main(void)

{

long distMeasured;

WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer

TA0CTL = TASSEL\_2 + MC\_1 + TAIE +ID\_3; // Timer A control set

to SMCLK, 1MHz and count up mode MC\_1

P4DIR |= TXD;

P4OUT |= TXD;

REFCTL0 &= ~REFMSTR; // Reset REFMSTR to hand over control to ADC12\_A

ref control registers

ADC12CTL0 |= ADC12SHT03 + ADC12ON + ADC12REFON; // Sampling time 32

cycles, ADC12 on and reference voltage for Temperature sensor is on

ADC12CTL1 |= ADC12SHP; // sampling timer

ADC12MCTL0 = ADC12SREF\_1 + ADC12INCH\_10; // Select internal Vref and A10

= temp sense i/p

ADC12CTL0 |= ADC12ENC ; // enable conversion

\_\_delay\_cycles(1000); // Wait for ADC Ref to settle

/\* Configure hardware UART \*/

UCA1CTL1 = UCSWRST; //Recommended to place USCI in reset first

P4SEL |= BIT4 + BIT5;

UCA1CTL1 |= UCSSEL\_2; // Use SMCLK

UCA1BR0 = 109; // Set baud rate to 9600 with 1.048MHz clock (Data Sheet

36.3.13)

UCA1BR1 = 0; // Set baud rate to 9600 with 1.048MHz clock

UCA1MCTL = UCBRS\_2; // Modulation UCBRSx = 2

UCA1CTL1 &= ~UCSWRST; // Initialize USCI state machine/\* if we were going to receive, we would also:

IE2 |= UCA1RXIE; // Enable USCI\_A1 RX interrupt

\*/

while(1)

{

ADC12CTL0 |= ADC12SC; // Sampling and conversion start

while (! (UCA1IFG & UCTXIFG)); // wait for TX buffer to be ready for

new data

distMeasured = ADC12MEM0;

distMeasured = distMeasured / 58;

TXByte = distMeasured;

UCA1TXBUF = TXByte;//Transmit TXByte;

// go to sleep, wait till timer expires to do another measurement.

\_delay\_cycles(100000);

}

}

Art (data processing):

* Watched pygame tutorials and read pygame documentation for information about the animation
  + <https://www.youtube.com/watch?v=AY9MnQ4x3zk&t=3877s>
  + <https://www.pygame.org/docs/>
* Write a loop using pygame that continuously updates the animation according to the data that is fed from the SR04. The timing of the python loop should be the same as the timing from the SR04.
* The calculation that is needed for where the eyes will be placed is done but taking the distance that the ultrasonic distance sensor outputs to determine where the person is relative to the eyes
* The image below shows the vectors of the eyes to the person

A drawing of a person with lines and arrows

Description automatically generated

* For each vector, there is an x and y component. The y component is constant and equal to the distance between the projector of eyes and the ultrasonic distance sensor:

A drawing of a person with a diagram

Description automatically generated

* The placement of each pupil is then determined via simple trigonometric calculations:

A drawing of a triangle

Description automatically generated

* However, the x-component of the vector must be calculated because the distance sensor only returns the distance between the person and the sensor, not the horizontal distance between the person and the eye
* Calculations are made by having an array indicating the placement of each eye. The initial position is subtracted from the distance measurement to get the x-component
* The line below is the python code for updating the angles of each eye:

positions = np.atan((distance – x0\_array) / y))

* Positions is an array of the angles of each eye in order from left to right. Distance is a float communicated from the ultrasonic distance detector to the MSP430 to the computer, divided by 58 for centimeters. y is a pre-defined constant that is the distance between the projector and the distance measurer.
* The current python program is shown below:

﻿#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

"""

main:

1. read data from MSP430 using serial

2. calculate array of vectors

3. update animation

"""

# import statements

import pygame

from sys import exit

import serial

import numpy as np

# read port (information coming from MSP430)

port = '/dev/tty.usbmodem11203'

# initiate pygame

pygame.init()

clock = pygame.time.Clock()

# set screen

screen = pygame.display.set\_mode((800, 400))

bg = pygame.Surface((800, 400)).convert()

bg.fill('White')

# not sure if this is the correct placement for the serial port

# with serial.Serial(port,9600,timeout = 0.050) as ser:

# print(ser.name)

# infinite loop

while True:

for event in pygame.event.get():

# if x out of game window, stop running code

if event.type == pygame.QUIT:

pygame.quit()

exit()

# get data from MSP430

data = ser.read(1)

if len(data) > 0:

distance = ord(data)

# update surfaces

screen.blit(bg, (0,0))

# display screen

pygame.display.update()

clock.tick(60)

* There are several more modifications that need to be done with the python program:

1. Vector calculations as described above need to be inserted
2. The order of serial reading vs pygame while loop need to be verified
3. Images of the eye positions need to be created and imported