SHARP GP2Y0A02YK0F Precision Distance Measurement

EE437 Report

Fall 2020

By

Rishi Zaveri

Abstract:

An embedded system measures distance using an infrared light beam. Infrared light is electromagnetic radiation that is longer in wavelength compared to visible light. The sensor should emit IR light and receive its reflection. A Raspberry Pi 4 and TI MSP432 device process the analog output from the sensor into precise distance measurements. The results will be displayed on an LCD display in decimal form and with a graph of distance over time. C++ and Python code will process the analog voltage data from the sensor into usable data.

Section A:

1. *Project Description:*

Key criteria for successful outcome:

- Precise distance measurement is computed using an IR sensor, TI MSP432 ADC device, and Raspberry Pi 3 or 4 with Adafruit PiTFT display.
- The program can display a dedicated GUI for measurement display and graphing.
- Analog signal output from sensor is processed into measurement data.
- Distance over time graph can actively update on the user interface along with current decimal value.
- Data can be recorded and stored in a file on the Raspberry Pi.

Significant components list:

- SHARP GP2Y0A21YK IR Sensor
- Raspberry Pi 4
- TI EXP-MSP432401R ADC Converter
- 10uF Capacitor
- Adafruit PiTFT Touchscreen 320x240 2.8”

2. *Constraint Analysis:*

The Raspberry Pi 3 or 4 would need to be used as the central embedded processor. The TI MSP432 board must convert analog signals from the sensor to digital. An LCD touchscreen must allow system functionality and calibration. C++ and Python can be used to program the software.

3. *Hardware Interface:*

A 2.8 inch PiTFT board connects to the first 26 pins of the Raspberry Pi. The Pi connects to the TI MSP432 board with the USB cable for the purpose of supplying power and receiving serial communication. Pins 21-23 on the TI board connect to the sensor using a breadboard. A 10uF capacitor stabilizes the power supply.

4. *Software Interface:*

The Energia IDE is used to program the TI board. The software digitally converts analog input and sends it through serial communication. Python is used to program the Pi with the help of Thonny IDE. From Python, the tkinter, serial, threading, and numpy libraries are used throughout the main code. Tkinter allows a GUI to be built. The GUI provides buttons, text display, and graph display. Threading allows the program to run the GUI while collecting and processing data from the TI board. Digital signal is converted to measurement data while a GUI displays this data.

5. ***Power Constraints and Efficiency:***

The SHARP IR sensor restricts the measurement distance to within 20cm to 150cm. The supply voltage required to run the sensor, as well as the TI and Pi boards is 5 volts each. The delay from the sensor measurement to analog output signal is at most 5.0ms. The time required for each measurement is at most 38.3ms with a tolerance of 9.6ms. The baud rate is held at 9600 bits/second for communication between the TI and Pi.

6. ***Cost Constraints:***

This product requires a Raspberry Pi 3 or 4 board, a TI EXP-MSP432P401 board, an IR precision sensor, and 2.8 inch PiTFT touchscreen.

- 4gb Raspberry Pi - $60
- SHARP IR Sensor - $10
- TI EXP-MSP432P401R - $20
- Adafruit PiTFT Touchscreen 320x240 2.8” - $35
- Breadboard and male-to-female pin connectors - $5

7. ***Component Selection Rationale:***

A breadboard and male to female connectors are used for circuit connections in creating the prototype. This gives flexibility in experimenting with circuit design. The PiTFT Touchscreen fits conveniently onto the Pi, providing a sufficient 2.8 inch display for the GUI. The Raspberry Pi 4 provides enough memory to run the program smoothly. The TI board provides effective ADC capabilities with computational abilities and multiple options for data transmission. The 10uF is recommended by the manufacturer for optimizing their sensor.

Section B:

Detailed project description:

1. ***Significant system components:***

Sharp IR Sensor GP2Y0A02YK0F: This device uses infrared light to produce analog signals which can be processed into short, precise measurements of distance between 20cm to 150cm.

Raspberry Pi 4 Model B: The memory, computational power, and GPIO this device offers allows for rapid prototyping with room for future implementations.

TI-EXPMSP432401R: The TI device offers its ADC property, computes data, and provides options for data transmission. The device is also programmable.

2.8 inch PiTFT Touchscreen LCD: The display is designed for compatibility with the Raspberry Pi 4. It allows for sufficient user interaction with the software.

2. ***Project Description:***

Step 1: Using pins 21-23 on the TI-MSP432 board, connect the SHARP IR Sensor GP2Y0A02YK0F for power, ground, and analog signal through a breadboard.

Step 2: Add a 10uF by-pass capacitor in parallel with the power supply and ground connections.

Step 3: Connect the TI board to a PC and, using Energia IDE, program the TI board to read analog input from pin 23 and setup serial communication. Proper connection and functionality of the sensor can be tested here.

Step 4: Connect the TI board to the Raspberry Pi 4 Model B with USB connector.

Step 5: Insert an SD card with Raspbian OS that can use python 3 libraries and Thonny IDE.

Step 6: Connect the Pi to a 5V power supply, HDMI monitor, keyboard, mouse, and wired or wireless internet.

Step 7: Using the terminal window, use the commands "sudo apt install python3-matplotlib" and "pip3 install numpy" and "sudo apt get install python-tk" to have access to these library resources.

Step 8: Create a python script using Thonny IDE which will run data collection, processing, and provide a GUI.

Step 9: Connect the PiTFT touchscreen to the Pi using the first 26 pins. Follow the procedure found in https://learn.adafruit.com/adafruit-pitft-28-inch-resistive-touchscreen-display-raspberry-pi/easy-install-2 to set up the Pi for the touchscreen.

Section C:

Additional Documents and Data Sheets

a. **Program Code:**

- **From Energia IDE for use with TI MSP432 Red Board:**

```
int analogPin = 23; // Pin 23 Analog Input
int analogValue = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  analogValue = analogRead(analogPin);
  Serial.println(analogValue);
  delay(500);
}
```

- **Main program using Python for use with the Raspberry Pi 4:**

```
import serial
import string
import tkinter as tk
import threading
import sys
import numpy as np
import matplotlib.pyplot as plt

from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import(FigureCanvasTkAgg, NavigationToolbar2Tk)

# Obtain sensor values, preprocess, and create formatted output to Shell
# This step will run as a thread which terminates on GUI closing
sensorValue = " "
voltageValue = 0.0
ser = serial.Serial('/dev/ttyACM0')

def sensing():
    while True:
        sensorValue = ser.readline()
        sensorValue = sensorValue.decode('utf-8').strip()
        print(sensorValue)
        voltageValue = int(sensorValue) * (5.0/1023.0)
        print("Voltage: " + str(round(voltageValue, 4)) + " Volts")
        cmDistance = 2*(10650.08 * pow(int(sensorValue),-0.935) – 10)
        print("Distance: " + str(round(cmDistance, 4)) + " cm")

        mmDistance = cmDistance * 10
        print("Distance: " + str(round(mmDistance, 4)) + " mm")

        inchDistance = cmDistance / 2.54
        print("Distance: " + str(round(inchDistance, 4)) + " inches")

        print(" ")

t1 = threading.Thread(target=sensing)
t1.daemon = True
t1.start()




# Create root of GUI using Tk() from tkinter library
root = tk.Tk()

# Add push buttons START, STOP, RECORD
userFrame = tk.LabelFrame(root, padx=0, pady=0)
userFrame.pack()

def startClick():
    print("start!")

def stopClick():
    print("stop!")

def recordClick():
    print("record!")

startButton = tk.Button(userFrame, text="Start", width=7, height=2, bg="green", command=startClick)
startButton.grid(row=0, column=0, padx=12)
stopButton = tk.Button(userFrame, text="Stop", width=7, height=2, bg="red", command=stopClick)
stopButton.grid(row=0, column=1, padx=12)
recordButton = tk.Button(userFrame, text="Record", width=7, height=2, bg="grey", command=recordClick)
recordButton.grid(row=0, column=2, padx=12)
```

```
# Add measurement options in cm, mm, and inch
# distFrame is a LabelFrame containing measureOptions and distFrame2
# distFrame2 is a LabelFrame containing distValue and distUnit
distFrame = tk.LabelFrame(root, padx=20, pady=0)
distFrame.pack()

clicked = tk.StringVar(distFrame)
clicked.set("cm  ")
measureOptions = tk.OptionMenu(distFrame, clicked, "cm  ", "mm ", "inch")
measureOptions.grid(row=0, column=0)

distFrame2 = tk.LabelFrame(distFrame, padx=50, pady=0)
distFrame2.grid(row=0, column=1)

distValue = tk.Label(distFrame2, width=7, text="--", padx=10)
distValue.grid(row=0, column=0)

distUnit = tk.Label(distFrame2, width=3, textvariable=clicked, padx=10)
distUnit.grid(row=0, column = 1)




# Plot results in distance vs time, configure y axis with appropriate measurement
graphFrame = tk.LabelFrame(root)
graphFrame.pack(expand=True)

fig = Figure(figsize=(5,4), dpi=100)
fig.add_subplot(111).plot([1,2,3], [4,5,1])

canvas = FigureCanvasTkAgg(fig, master=graphFrame)
canvas.draw()
canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)




# Run GUI and terminate daemon thread on exit
root.geometry("380x280") # root.geometry("320x240")
root.mainloop()

sys.exit()
```

```
#===============
# Notes/Example Code:
#
# Use append function for graphing
# matplotlib library
# numpy library
# sudo apt install python3-matplotlib
# pip3 install numpy
# sudo apt get install python-tk
#
# import tkinter
# root = tkinter.Tk()
#
# myLabel1 = tk.Label(root, text="Hello World") # Add text
# myLabel1.grid(row=0, column=0) # place myLabel in most convenient spot of root widget
#
# entry = Entry(root, width=50, borderwidth=5) # Add text box
# entry.pack() # Place entry in most convenient spot of root widget
# entry.get() # get text from the text box
```

b. Circuit Diagram

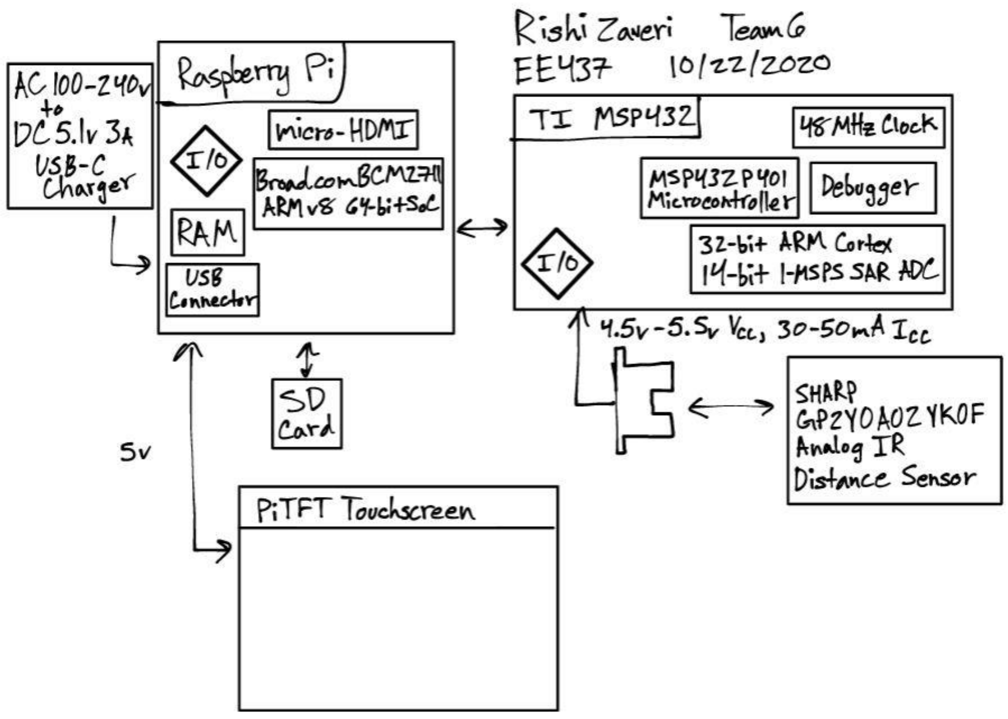
Rishi Zaveri Team6
EE437 10/22/2020
AC 100-240v to DC 5.1v 3A USB-C Charger
Raspberry Pi
I/O
micro-HDMI
Broadcom BCM2711 ARMv8 64-bit SoC
RAM
USB Connector
SD Card
5v
PiTFT Touchscreen
TI MSP432
48 MHz Clock
MSP432P401 Microcontroller
Debugger
I/O
32-bit ARM Cortex 14-bit 1-MSPS SAR ADC
4.5v-5.5v Vcc, 30-50mA Icc
SHARP GP2Y0A02YK0F Analog IR Distance Sensor

Image 2. Device Connection Diagram and Specifications

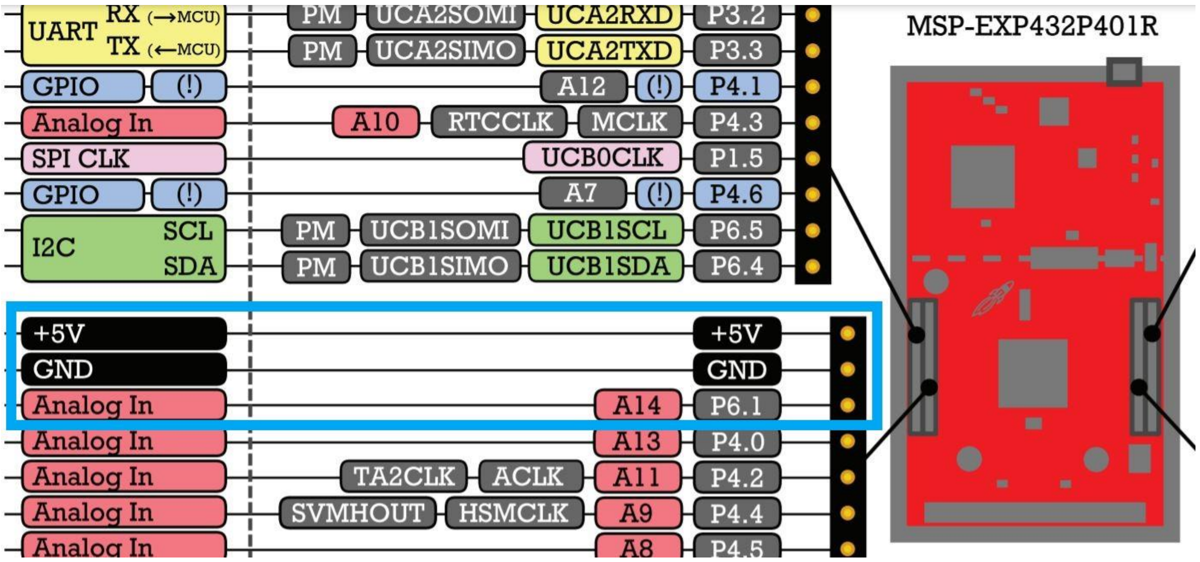

Image 3. Device Pins for TI MSP432

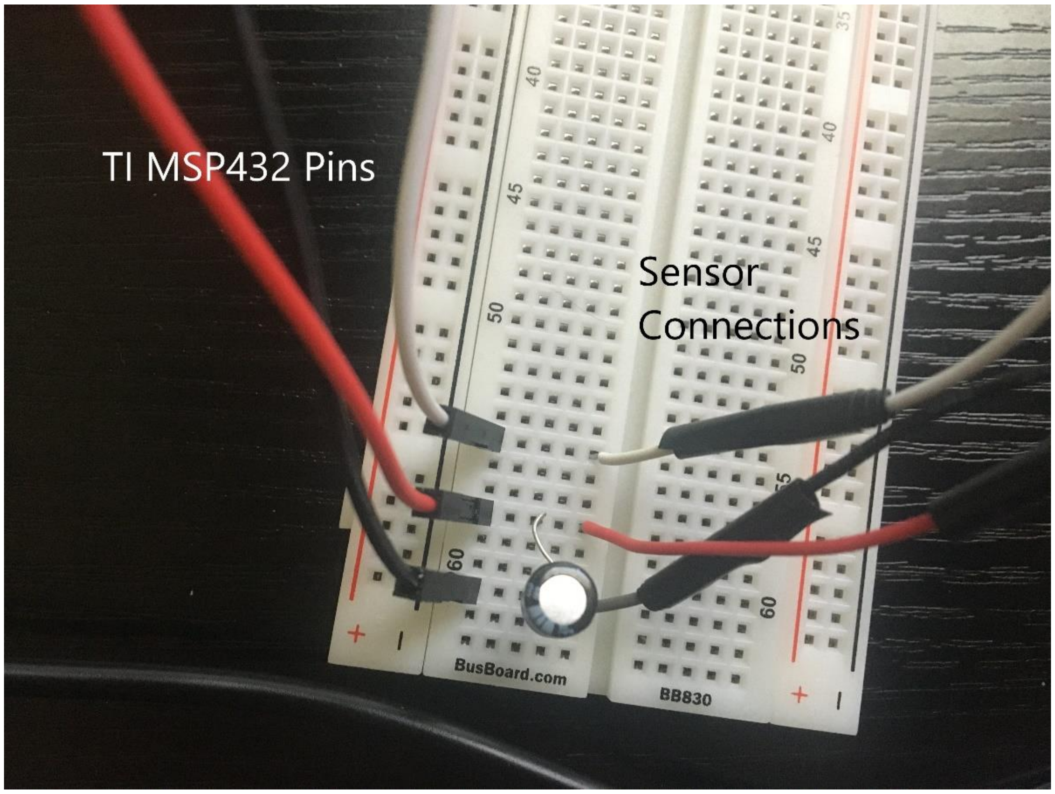

Image 4. Circuit Connections

c. Libraries:

Serial - https://pyserial.readthedocs.io/en/latest/pyserial_api.html

String - https://docs.python.org/3/library/string.html

Tkinter - https://docs.python.org/3/library/tk.html

Threading - https://docs.python.org/3/library/threading.html

Sys - https://docs.python.org/3/library/sys.html

Numpy - https://numpy.org/doc/stable/contents.html

Matplotlib - https://www.kite.com/python/docs/matplotlib

d. References:

https://www.digikey.com/htmldatasheets/production/9651/0/0/1/gp2y0a02yk0f.html

https://energia.nu/reference/en/language/functions/communication/serial/

https://www.youtube.com/watch?v=YXPyB4XeYLA&t=12021s

https://www.tutorialspoint.com/python/python_gui_programming.htm

https://www.geeksforgeeks.org/how-to-create-a-new-thread-in-python/