Sink N' Save

Smart Faucet

Cory Clark, Mark Olivas, Quinton Jamora

Sponsored By:

PME Services, Inc. & Sustainability Decathlon

Sustainability Decathlon

ORANGE COUNTY

Introduction

Our Team

Cory took lead of **Electrical Engineering** and plumbing aspect of this project, this includes all of the circuitry along with all of the piping our sink contains.
Mark and Quinton led the **Embedded** and **Software Engineering** aspect of the project. This includes configuring our microprocessor, machine learning and other programming.

Our Goal

We wanted to create a device that could save as much water as possible. Using our knowledge of electronics and computer systems, we believe the use of machine learning and object detection would be a great way to **control water usage**, considering the vast amount of objects that we use under our sinks today.

Inspiration

Automation

Firstly, we wanted to build something **automated** that could potentially help us in our daily lives. This includes no physical or manual touch to the device.

Environment

Next we believed that we could create something that could be beneficial to not only ourselves, but the **environment**. In this case it was water preservation.

Collaboration

As we learned more about this project, Dr. George mentioned that our device could potentially be implemented in a “Smart Home”, where we would **collaborate** with Civil Engineers.
(Sustainability Decathlon)

Requirements, Specifications, Design

Requirements

There are many different parts and components that we needed to build this project, considering the compatibility when working with machine learning and electronics.

Design

For our main design, we repurposed an old sink and used our own piping combined with water pumps to push water out of the faucet. Our camera used for object detection is mounted on top of the faucet.

Specifications

We decided to use a Raspberry Pi 4b as our microprocessor to act as the brain of our project and send signals to our Relays (4x) that would control the Solenoids (4x).

Key Functionalities

When the solenoids receive a signal from the Raspberry Pi, they will open, therefore releasing hot or cold water, and close when the signal is terminated. This signal can range from object or motion detection.

Experimentation // Challenges

01

Originally, we were going to run our program using a Laptop and Arduino, but we decided to switch to a Raspberry Pi, because it would be more efficient for our design.

02

The first sink prototype we made had a base that was handmade out of wood with the help of Bobby, for our final design we decided to purchase an all in one sink compartment.

03

Our design uses a combination of OpenCV and TensorFlow for its uses in Object Recognition all in Python, considering it contains various libraries to use for programming.

04

Using Google's Teachable Machine to create an Object Recognition Library. Many object libraries already exist, but teaching our own objects would be more beneficial for our design.

05

The biggest challenge we faced was definitely coding the sink. From including the TensorFlow and Object library to creating the Timing Mechanisms that the sink uses between each object and condition.

06

Other challenges we faced were making decision on different features we wanted to add such as Soap Dispenser, Motion detection, and even Voice Recognition.

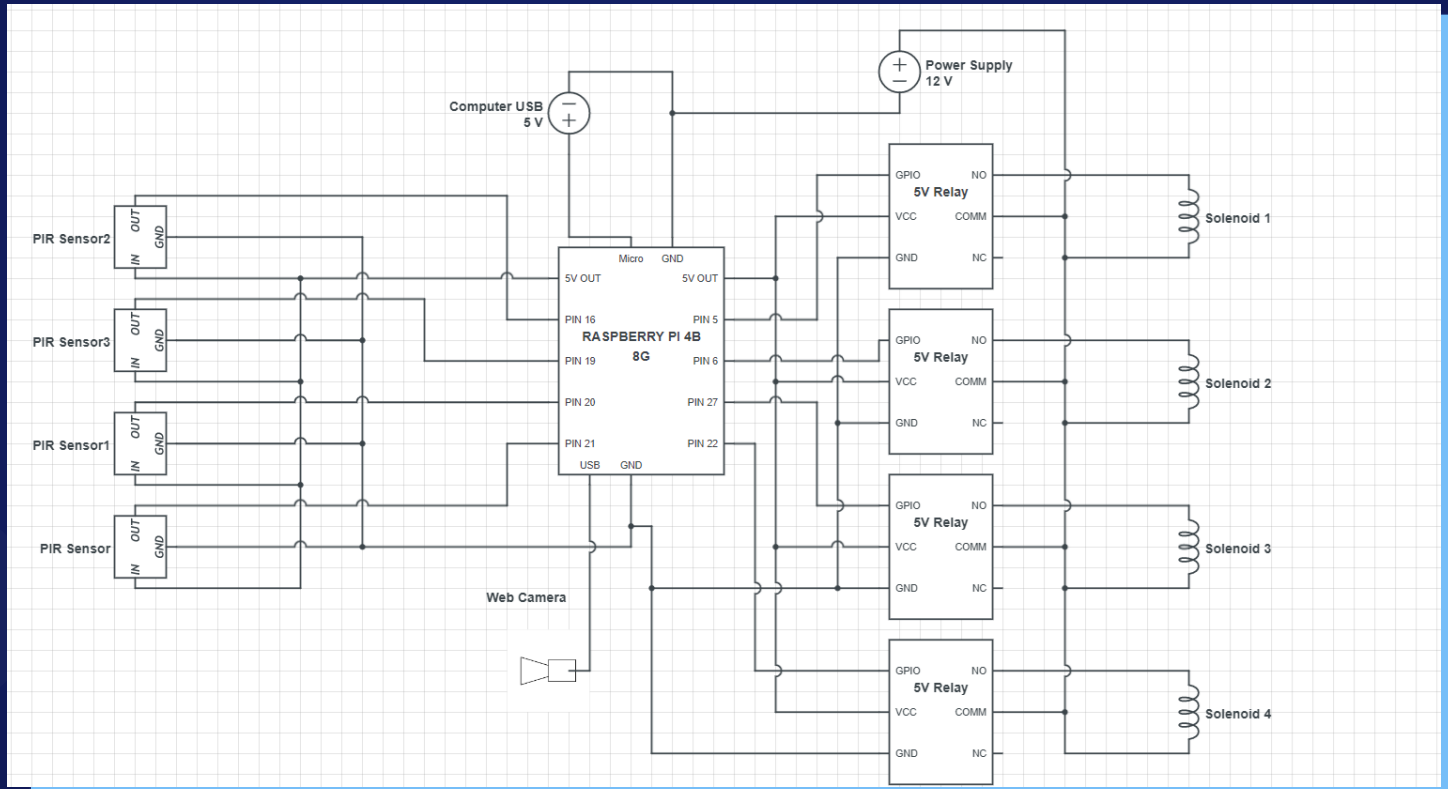
First Prototype

Updated Prototype

Electronics Setup

Video Demonstration

Schematic


Power Supply
12 V
Computer USB
5 V
PIR Sensor2
PIR Sensor3
PIR Sensor1
PIR Sensor
OUT
GND
IN
Micro
GND
5V OUT
5V OUT
PIN 16
PIN 5
RASPBERRY PI 4B
8G
PIN 19
PIN 6
PIN 20
PIN 27
PIN 21
PIN 22
USB
GND
Web Camera
GPIO
NO
5V Relay
VCC
COMM
GND
NC
Solenoid 1
Solenoid 2
Solenoid 3
Solenoid 4

Results // Discussion

Relays ✓

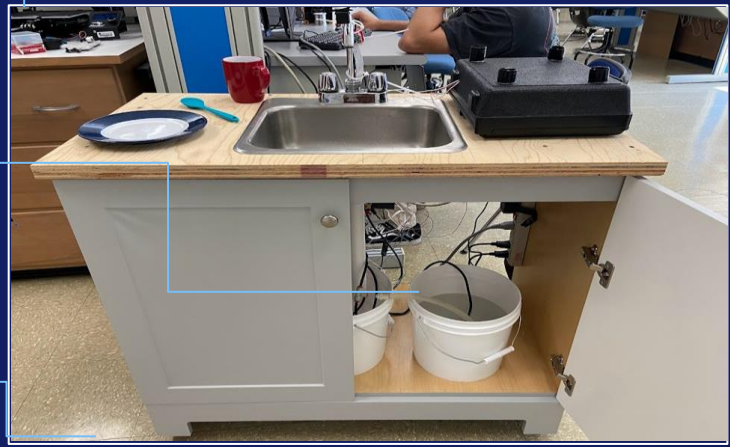
Solenoids ✓

Water Flow ✓

Motion Sensor ✓

Object Detection ✓

ALL TOGETHER ✓ ✓ ✓ ✓ ✓



The code

```
import cv2
import time
import RPi.GPIO as GPIO
import numpy as np
import TensorFlow as tf
from keras.models import
load_model


# GPIO pins initialization
# Specifically GPIO Pins
solenoidHot2 = 5
solenoidCold1 = 27
solenoidCold2 = 22
solenoidHot1 = 6
motionFreeFlow = 21
motionHot = 19
motionCold = 16
motionWaterOff = 20
tempMotion = 0     <------

GPIO.setmode(GPIO.BCM)
GPIO.setup(
    [solenoidHot2, solenoidHot1,
solenoidCold1, solenoidCold2],
    GPIO.OUT,
    initial=0
)
GPIO.setup(
    [motionFreeFlow, motionHot,
motionCold, motionWaterOff],
    GPIO.IN
)
GPIO.setwarnings(False)  .  .  .
```

```
# Load the model
model =
load_model("/home/pi/Desktop/Objec
t_Detection_Files/keras_model.h5",
compile=False)

# Load the labels
class_names =
"/home/pi/Desktop/Object_Detection
_Files/labels.txt"
with open(class_names, 'r') as f:
    class_name =
f.read().rstrip("\n").split("\n")

# CAMERA can be 0 or 1 based on
the default camera of your
computer
camera = cv2.VideoCapture(1) **

# Define a dictionary to store the
timestamps of detected objects
object_timestamps = {}  <------

def allOff():
    GPIO.output(solenoidCold1, 0)
    GPIO.output(solenoidCold2, 0)
    GPIO.output(solenoidHot1, 0)
    GPIO.output(solenoidHot2, 0)

def allOn():
    GPIO.output(solenoidCold1, 1)
    GPIO.output(solenoidCold2, 1)
    GPIO.output(solenoidHot1, 1)
    GPIO.output(solenoidHot2, 1)
```

```
def process_frame(image):
    # Resize the raw image into (224-height,224-width)
pixels
    image = cv2.resize(image, (224, 224),
interpolation=cv2.INTER_AREA)

    # Show the image in a window
    cv2.imshow("Webcam Image", image)
# Make the image a numpy array and reshape it to the
models input shape.
    image = np.asarray(image,
dtype=np.float32).reshape(1, 224, 224, 3)

    # Normalize the image array
    image = (image / 127.5) - 1

    # Predict the model
    prediction = model.predict(image)
    index = np.argmax(prediction)
    confidence_score = prediction[0][index]

    return index, confidence_score
#-------labels from labels.txt ----------
empty = 0, spoon = 1, cup = 2,
plate = 3, hands = 4

while True:
    # Grab the web camera's image.
    ret, image = camera.read()

    if not ret:
        break

    # Perform object detection
    index, confidence_score = process_frame(image)

    current_time = time.time()

    waterOff = GPIO.input(20)
    freeFlow = GPIO.input(21)
    cold = GPIO.input(16)
    hot = GPIO.input(19)
```

The code cont.

```
if waterOff == 1: #Motion Detection Off
        tempMotion = 1
        print("Motion Water Off")
        allOff()
        time.sleep(3)
    elif freeFlow == 1:
            allOn()    <----------
        tempMotion = 1
        print("Motion Water All On")
        time.sleep(3)
    elif cold == 1:
        allOff()
        tempMotion = 2
        print("Temp Motion 2")
    elif hot == 1:
        allOff()
        tempMotion = 3
        print("Temp Motion 3")

if tempMotion == 2: #Motion Detection Cold
        print("Motion Water Cold")
        if index == spoon and not detected_spoon:
            print("Cold spoon")
            if index not in object_timestamps or
(current_time - object_timestamps[index]) > 10:
                GPIO.output(solenoidCold1, 1)
                time.sleep(3)
                allOff()
                detected_spoon = True
                object_timestamps[index] = current_time
```

```
        elif index == cup and not detected_cup:
            print("Cold cup")
            if index not in object_timestamps or
(current_time - object_timestamps[index]) > 10:
                GPIO.output(solenoidCold1, 1)
                GPIO.output(solenoidCold2, 1)
                time.sleep(3)
                allOff()
                detected_cup = True
                object_timestamps[index] = current_time

        elif index == plate and not detected_plate:
            print("Cold plate")
            if index not in object_timestamps or
(current_time - object_timestamps[index]) > 10:
                GPIO.output(solenoidCold1, 1)
                GPIO.output(solenoidCold2, 1)
                GPIO.output(solenoidHot1, 1)
                time.sleep(3)
                allOff()
                detected_plate = True
                object_timestamps[index] = current_time
        elif index == hands and not detected_hands:
            print("Cold hands")
            if index not in object_timestamps or
(current_time - object_timestamps[index]) > 10:
                GPIO.output(solenoidCold1, 1)
                GPIO.output(solenoidCold2, 1)
                time.sleep(3)
                allOff()
                detected_hands = True
                object_timestamps[index] = current_time
```

The code cont..

```
elif tempMotion == 3: #Motion Detection Hot
        print("Motion Water Hot")
        if index == spoon and not
detected_spoon:
            print("Hot spoon")
            if index not in object_timestamps or
(current_time - object_timestamps[index]) > 10:
                GPIO.output(solenoidCold1, 1)
                time.sleep(3)
                allOff()
                detected_spoon = True
                object_timestamps[index] =
                                current_time
        elif index == cup and not detected_cup:
            print("Hot cup")
            if index not in object_timestamps or
(current_time - object_timestamps[index]) > 10:
                GPIO.output(solenoidCold1, 1)
                GPIO.output(solenoidCold2, 1)
                time.sleep(3)
                allOff()
                detected_cup = True
                object_timestamps[index] =
                                current_time
elif index == plate and not detected_plate:
            print("Hot plate")
            if index not in object_timestamps or (current_time -
object_timestamps[index]) > 10:
                GPIO.output(solenoidCold1, 1)
                GPIO.output(solenoidCold2, 1)
                GPIO.output(solenoidHot1, 1)
                time.sleep(3)
                allOff()
                detected_plate = True
                object_timestamps[index] = current_time
        elif index == hands and not detected_hands:
            print("Hot hands")
            if index not in object_timestamps or (current_time -
object_timestamps[index]) > 10:
                GPIO.output(solenoidCold1, 1)
                GPIO.output(solenoidCold2, 1)
                time.sleep(3)
                allOff()
                detected_hands = True
                object_timestamps[index] = current_time
            # Timing Delay..
    if detected_spoon or detected_plate or detected_cup or
                                                    detected_hands:
        cv2.waitKey(5000)
        detected_spoon = False
        detected_plate = False
        detected_cup = False
        detected_hands = False
        camera.read()
```

Total Budget

Item	Price
Raspberry Pi 4b	159.00
Sink w/ compartment	250.00
Electrical components	275.00
Plumbing	150.00
Other	200.00
TOTAL:	**$1,034.00**
REALISTICALLY:	**$1,800.00**

Conclusion

Overall, we all gained a wealth of knowledge working hard to design and develop a solution which contains different aspects of electrical, embedded, and software engineering. Our design could potentially be helpful to not only our everyday lives, but to the environment as well. We are not only more confident, but excited to continue our journeys as engineers in the future.

Thank You

Any questions?