

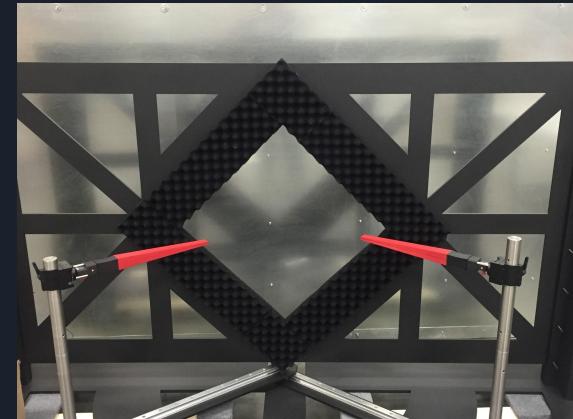
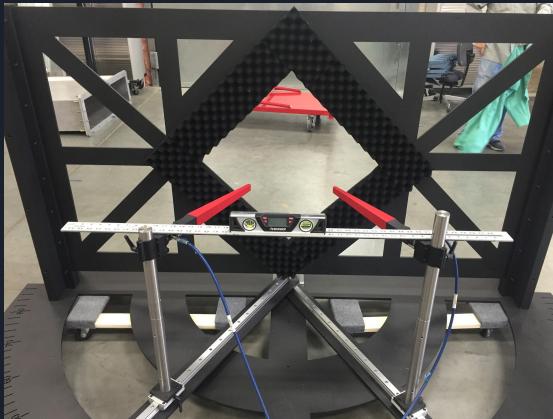
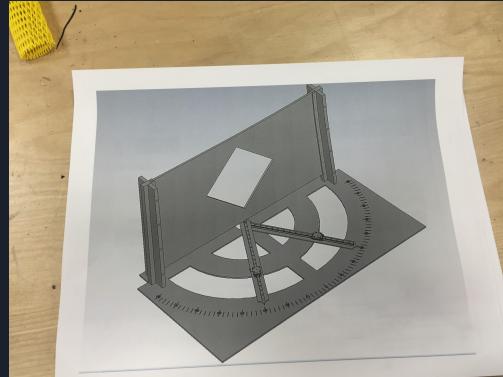


Projects and Research

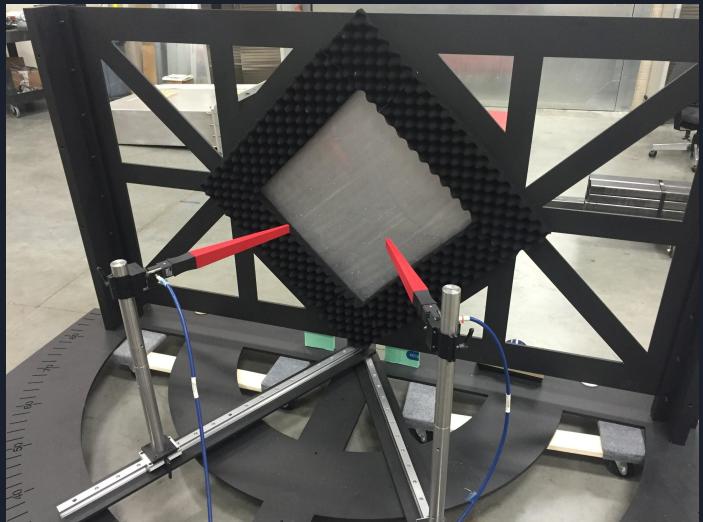
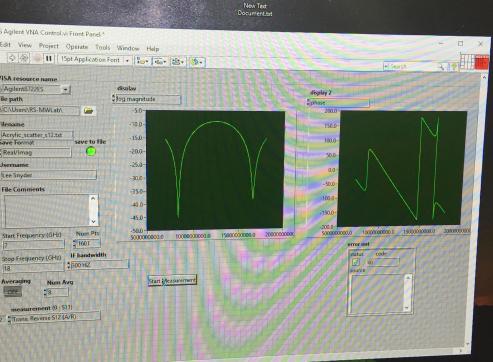
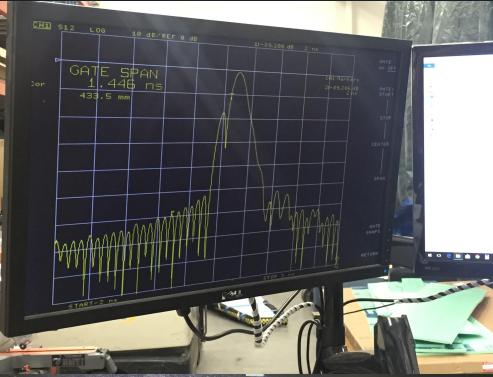
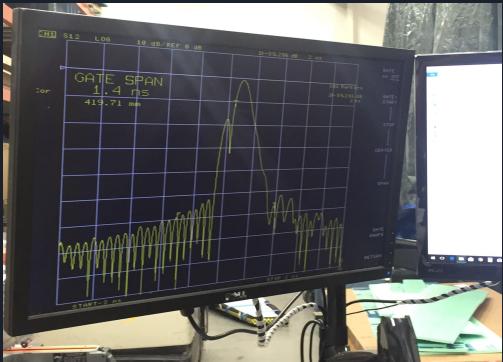
Rick Hubbard



Scatterometer Custom Build Phase



Scatterometer Testing Phase



Code Excerpt from Processing

```
//displays temperature probe reading
textFont(f,75);
fill(255);
text(value2 + "°F",130,200);
//textFont(f,16);
//fill(255);
//text("The current temperature is:", 100,180);
//system is heating...display heating
if (value2 < value1)
{
    textFont(f,32);
    fill(204,0,0);
    text("Heating", 140,250);
}
//system is idle...display idle
if (value1 == value2 || (value2 > value1 && value2 < value1+5))
{
    fill(255);
    text("Idle", 175,290);
}
//system is cooling...display cooling
if (value2 > value1+5)
{
    textFont(f,32);
    fill(255);
    text("Cooling", 145,250);
}
```

Figure 5: Part of the Processing code to match the digital interface colors with that of the LEDs.

Arduino Microcontroller Project

Table 1: Components used in KTR System

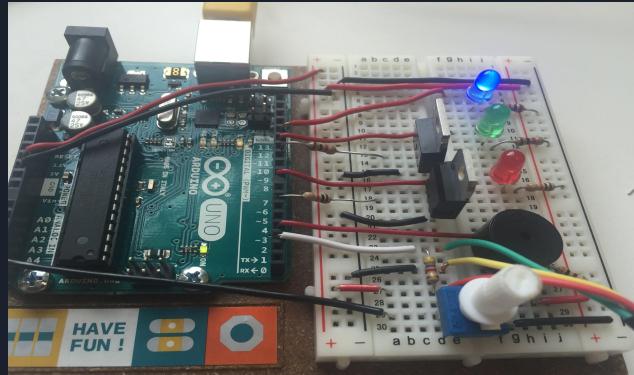
Component	Quantity
LEDs	3 (red, blue, and green)
1 kΩ Resistors	3
360 Ω Resistor	1
470 Ω Resistor	1
Piezo Speaker	1
Fan	1
Stemless wine glass	1
DS18B20 Digital Thermometer	1
20 ohm Power Transfer Resistor (heater)	2
Potentiometer	1
Transistors	2
12 V 0.5 A Fan	1
Arduino Uno Microcontroller	1
Processing 3.3.3	1
Digital Interface	1
Wires	Significant amount



Code Excerpt from Arduino

```
sensors.requestTemperatures();
//compare temperature from probe to set temperature
//if temperature is less than set temperature turn on heater
while(sensors.getTempFByIndex(0) < (analogRead(potPin)/3))
{
    //print the temperature and set temperature to serial port for processing to use
    //set temperature probe to an integer
    temp = sensors.getTempFByIndex(0);
    Serial.print(analogRead(A5)/3);
    Serial.print(",");
    Serial.println(temp);
    //turn on Red LED indicating heater is on
    digitalWrite(HeatingPin, HIGH);
    //set pin to transistor to high to turn on heater
    digitalWrite(powerPin, HIGH);
    //request temperature from probe again for comparison
    sensors.requestTemperatures();
}
```

Figure 4: Part of the temperature code



Java Computer Learning Tic Tac Toe Game

```
Welcome to Tic Tac Toe
This is a one player game against a computer
At the beginning of each of your turns you will be asked to enter a number 0-8 corresponding to which square you would like to take.
The board is numbered as follows:
0 | 1 | 2
-----
3 | 4 | 5
-----
6 | 7 | 8
Now lets get started
|
|
-----
|
|
-----
|
|
Please enter a number 0-8 to indicate which square you'd like to claim: 4
The computers best move is: 8
|
|
-----
| X |
-----
|
| 0
Please enter a number 0-8 to indicate which square you'd like to claim: 0
The computers best move is: 6
X | 
-----
| X |
-----
0 | | 0
```

Wireshark Wireless Data Interception Project

The screenshot shows two main windows from the Wireshark application.

Left Window (Main View): This window displays a list of captured network frames. The frames are color-coded by protocol: pink for ICMP. The columns include No., Time, Source, Destination, Protocol, Length, and Info. A pink-highlighted row represents frame 301, which is an ICMP Echo request (ping). The Info column for this frame shows the payload: "70 Time-to-live exceeded (Time to live exceeded in transit)".

Right Window (Details View): This window provides a detailed analysis of the selected ICMP frame (Frame 301). The title bar says "Internet Control Message Protocol". The details pane shows the following fields:

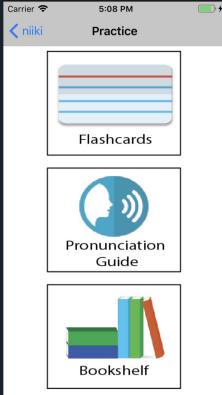
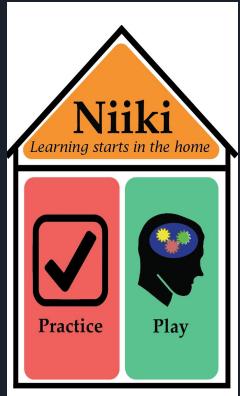
- Type: 8 (Echo (ping) request)
- Code: 0
- Checksum: 0xf7bd [correct]
- [Checksum Status: Good]
- Identifier (BE): 1 (0x0001)
- Identifier (LE): 256 (0x0100)
- Sequence number (BE): 65 (0x0041)
- Sequence number (LE): 16640 (0x4100)

A yellow-highlighted section at the bottom states "[No response seen]".

Bottom Status Bar: The status bar at the bottom of the main window indicates "Packets: 86189 Displayed: 86 (0.1%)".

Bottom Right Corner: A small window titled "Frame 301" shows the raw hex and ASCII data of the selected ICMP frame.

Niiki Language App (iPhone and Android)



Symbol	As in Miami-Peoria	As in English
p	pakaani (nut)	paper
t	tawaani (tree)	tie
k	akooka (frog)	keep
c	aciwi (hill)	church
s	sakimia (mosquito)	see
š	šooli (money)	show
h	neehi (and)	ahead
m	miimia (pigeon)	mom
n	niila (I, me)	no
l	piloohna (child)	leave
w	waapiki (it is white)	wish
y	yaalanwi (five)	yarn
a	aniwka (squirrel)	pot
aa	waawi (egg)	fall
e	alemwa (dog)	bet
ee	neepika (he dies)	made
i	nipi (water)	big
ii	niiswi (two)	see

Bookshelf

Ehoni Klintoohki Pyawaaci Myaamiaki Start/Stop mihini myaamiaki nipinkonci saakadweewiiki, eehoni saakaciweewaci 'saakineewonki' itamenki.

nipinkonci neweweeyohsaaki moohkiicki, 'peemitanahkwahki sakaahtweeo', iiliticiki.

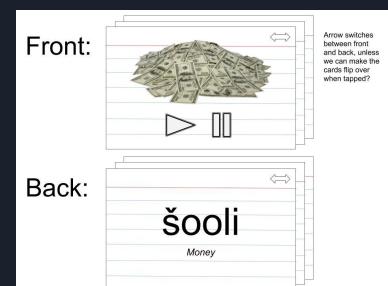
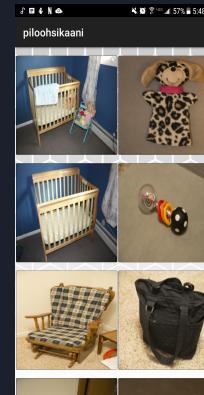
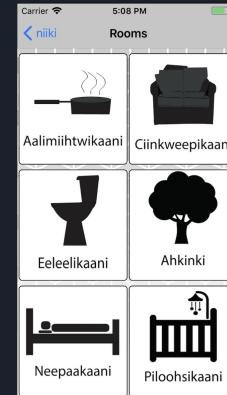
neehi akakaciweeki, noonki niyahai eeminooceoeki, niyanci maacilkaaci, minooteni neekatanki.

kapotive nikoli apweyeaya, apweyeaya kwiitaki miibaaheenaki neewacki saakineeyenki, naashpa-hes naapi ilataaweeeki ilataaweeyarki, neehi-hsa weentaaawaci.

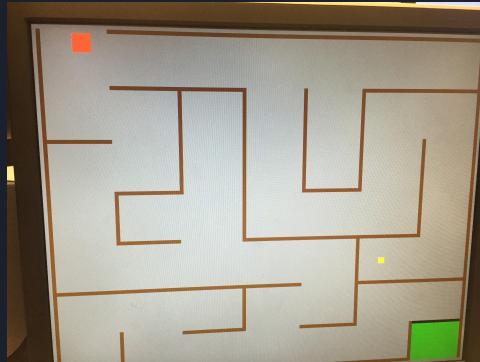
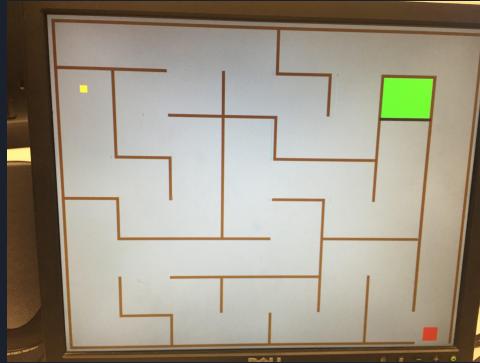
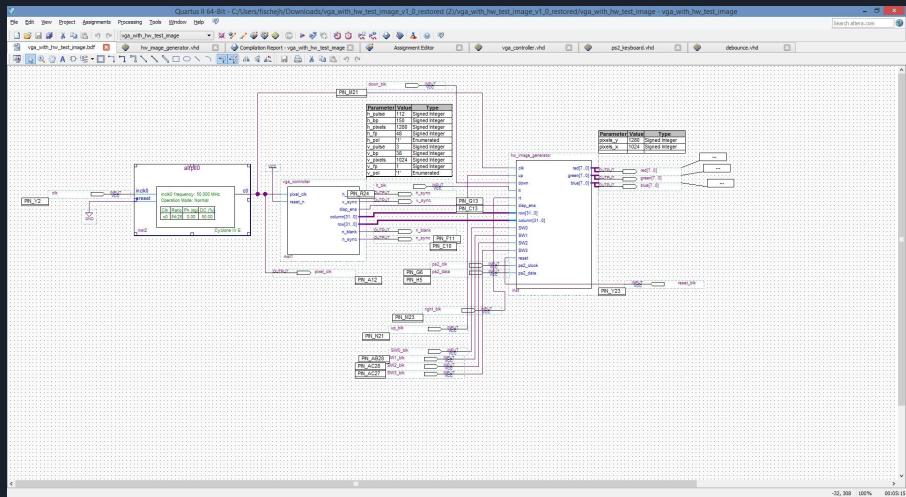
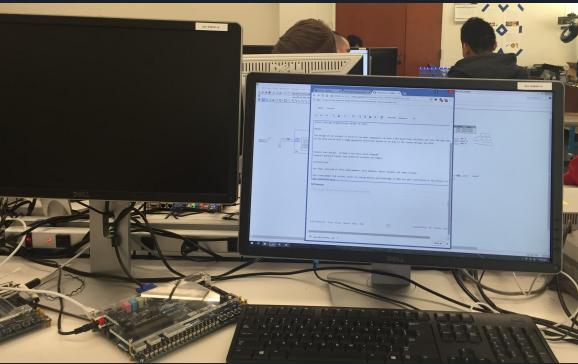
mathohkohsenaaakan ilaacki ina mihtoseenaki, moohci ninkheelmaahsoo weencinaakoswaadi, moohci awweyaki khieelmaawataa eehi ayawaci.

onini nikkii iismiwaaci, ninky seekaahkweeta amihsall waapankhikwa, ceekii mihtoseenaki kyosaki eelaamittaniki.

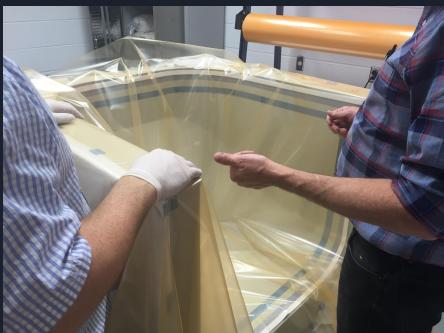
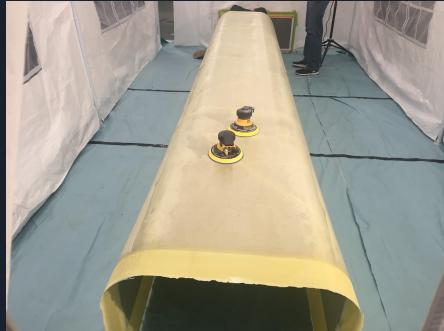
sipiwi 'saakieesisipiwi' weentanki, eehoni saakaciweewaci. linini wiyoontonki nisi weentanki



VHDL FPGA Maze Project

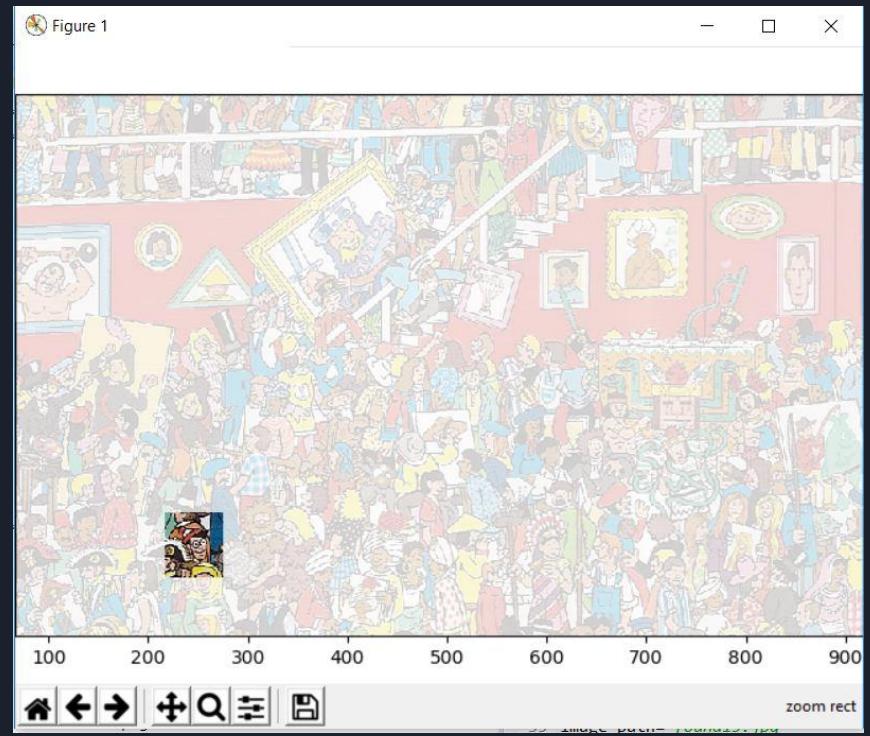


RF Composite Radome





Find Waldo Computer Vision Program



Tensorflow and Python Code

```
from matplotlib import pyplot as plt
import numpy as np
import sys
import tensorflow as tf
import matplotlib
from PIL import Image
import matplotlib.patches as patches
import cv2
import tkinter as tk
from tkinter import ttk

cap = cv2.VideoCapture(0)

while(cap.isOpened()): # check !
    # capture frame-by-frame
    ret, frame = cap.read()

    if ret: # check ! (some webcam's need a "warmup")
        # our operation on frame come here
        rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2BGRA)

        # Display the resulting frame
        cv2.imshow('frame', rgb)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            out = cv2.imwrite('capture.jpg', frame)
            break

    # When everything is done release the capture
cap.release()
cv2.destroyAllWindows()

model_path = 'frozen_inference_graph.pb'
image_np = np.array('found13.jpg')

def draw_box(box, image_np):
    #expand the box by 50%
    box *= np.array([- (box[2] - box[0])/2, -(box[3] - box[1])/2, (box[2] - box[0])/2, (box[3] - box[1])/2])

    fig = plt.figure()
    ax = plt.axes(fig, [0., 0., 1., 1.])
    fig.add_axes(ax)

    #draw blurred boxes around box
    ax.add_patch(patches.Rectangle((0,0),box[1]*image_np.shape[1], image_np.shape[0], linewidth=0,edgecolor='none',facecolor='w',alpha=0.8))
    ax.add_patch(patches.Rectangle((box[3]-image_np.shape[1],0), image_np.shape[1], image_np.shape[0], linewidth=0,edgecolor='none',facecolor='w',alpha=0.8))
    ax.add_patch(patches.Rectangle((box[1]*image_np.shape[1],0),(box[3]-box[1])*image_np.shape[1], image_np.shape[0], linewidth=0,edgecolor='none',facecolor='w',alpha=0.8))
    box[0]*image_np.shape[0], linewidth=0,edgecolor='none',facecolor='w',alpha=0.8))

    ax.add_patch(patches.Rectangle((box[1]*image_np.shape[1],box[2]*image_np.shape[0]),(box[3]-box[1])*image_np.shape[1], image_np.shape[0], linewidth=0,edgecolor='none',facecolor='w',alpha=0.8))

    return fig, ax
```

```
while(cap.isOpened()): # check !
    # capture frame-by-frame
    for t in range(0,250):
        ret, frame = cap.read()

    if ret: # check ! (some webcam's need a "warmup")
        # our operation on frame come here
        rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2BGRA)

        # Display the resulting frame
        #for t in range (0, 20000):
        cv2.imshow('frame', rgb)

    name = "frame%d.jpg"%count
    cv2.imwrite(name, frame)
    count += 1
    cv2.waitKey(1)
    break
```

```
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(model_path, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

def load_image_into_numpy_array(image):
    (im_width, im_height) = image.size
    return np.array(image.getdata()).reshape(
        (im_height, im_width, 3)).astype(np.uint8)

with detection_graph.as_default():
    with tf.Session(graph=detection_graph) as sess:
        image_np = load_image_into_numpy_array(Image.open(image_path))
        image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')
        boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
        scores = detection_graph.get_tensor_by_name('detection_scores:0')
        classes = detection_graph.get_tensor_by_name('detection_classes:0')
        num_detections = detection_graph.get_tensor_by_name('num_detections:0')
        # Actual detection
        (boxes, scores, classes, num_detections) = sess.run(
            [boxes, scores, classes, num_detections],
            feed_dict={image_tensor: np.expand_dims(image_np, axis=0)})

    if scores[0][0] < 0.1:
        sys.exit('Wally not found :(')

    print('Wally found')

    win = tk.Tk()
    win.title("Waldo Found")
    win.geometry("850x500")
    frame = tk.Frame(win, padding="100 300 10 10")
    frame.pack(fill=tk.BOTH, expand=True)
```

Object Detection Results

Program successfully detected Wally in the included 4 evaluation images and was also able to successfully find Wally in an additional 17 images for a total of 21 images.

