ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΤΜΗΜΑ
ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εργασία Εξαμήνου

Ενσωματωμένα Συστήματα

“Πυργίσκος Αυτόματης Παρακολούθησης”

Ονοματεπώνυμο: Ηλίας Σταθάκος ΑΜ: 2017

Προσοχή

Στο παρόν έγγραφο παρουσιάζονται οι οδηγίες για την υλοποίηση της συγκεκριμένης εργασίας. Οποιαδήποτε διαφοροποίηση τόσο στο υλικό όσο και στο λογισμικό, μπορεί να επιφέρει διαφορετικά αποτελέσματα από αυτά που παρουσιάζονται.

Εισαγωγή

Η συγκεκριμένη εργασία δημιουργήθηκε στα πλαίσια του μαθήματος Ενσωματωμένα Συστήματα το έτος 2025, υπό την επίβλεψη του Δρ. Μηνά Δασυγένη. Αφορά έναν πυργίσκο αυτόματης παρακολούθησης βασισμένο στο μοντέλο posenet, ένα αρκετά έμπιστο και διαδεδομένο μοντέλο βαθίας μάθησης (Deep learning). Ο Πυργίσκος διαθέτει δυνατότητα αδιάκοπης περιστροφής γύρω από τον κατακόρυφο άξονα του (άξονας Z) και δυνατότητα περιστροφής έως 75° γύρω από τον οριζόντιο άξονα (άξονα Y), ο οποίος είναι κάθετος στον Z και παράλληλος προς το νοητό ευθύγραμμο τμήμα AB που ορίζει τη μεγάλη πλευρά της βάσης του. Η αδιάκοπη περιστροφή στον άξονα Z επιτυγχάνεται περιστρέφοντας όλο το σώμα του πυργίσκου, μαζί με τα εξαρτήματα τροφοδοσίας του. Επίσης κατά την περιστροφή στον άξονα Y περιστρέφεται και η κάμερα. Η περιστροφή της κάμερας τόσο στον άξονα Z όσο και στον άξονα Y, δημιουργεί ένα σύστημα κλειστού βρόγχου(Closed-Loop-System), το οποίο από μόνο του επιτρέπει την χρήση ελεγκτών (πχ PID) για διόρθωση σφαλμάτων, χωρίς να χρειάζεται δηλαδή η χρήση κωδικοποιητών (encoder), για εξακρίβωση του λάθους κατά τις περιστροφές. Τέλος χρησιμοποιείται μια μέθοδο εξακρίβωσης της απόστασης από την κάμερα για εναλλαγή προσώπων. Η μέθοδος αυτή δεν βασίζεται σε depth tracking, αλλά σε μέτρηση του μεγέθους του προσώπου από την καταγραφή του καρέ(frame), πράγμα που μπορεί να το κάνει αναξιόπιστο σε μερικές περιπτώσεις. Βέβαια στα

πλαίσια του ελέγχου δεν εντοπίστηκε κάποιο έντονο σφάλμα στην εναλλαγή προσώπων με αυτή την μέθοδο.

Απαιτήσεις υλοποίησης

Στην συγκεκριμένη ενότητα παρουσιάζονται οι απαιτήσεις σε υλικό και λογισμικό για την υλοποίηση της εργασίας. Η συγκεκριμένη εργασία χωρίζεται σε 2 βασικά κομμάτια, το σύστημα κίνησης και το σύστημα παρακολούθησης, καθένα με το δικό του υλικό και λογισμικό.

Υλικό

<u>Σύστημα κίνησης:</u>

Για την περιστροφή στον άξονα Z, χρησιμοποιήθηκε ο βηματικός κινητήρας 42BYGHW804 από την σειρά 42BYGHW της wantai motors, ο οποίος ανήκει στο πρότυπο NEMA 17, ικανός να μετακινήσει το σώμα του πυργίσκου. Για την οδήγηση του βηματικού χρησιμοποιήθηκε το stepper driver A4988, το οποίο παρά τους περιορισμούς που έχει είναι μια καλή φθηνή επιλογή με δυνατότητα micro stepping, που χρησιμοποιείται σε αρκετά DIY project ρομποτικής.

Για την περιστροφή στον άξονα Y, χρησιμοποιήθηκε ο σερβοκινητήρας HS-645MG της HITEC, ένας κινητήρας υψηλής ροπής με μεταλλικά γρανάζια, ιδανικός για την μετακίνηση της κάμερας

Για τον χειρισμό του βηματικού κινητήρα και του σερβοκινητήρα χρησιμοποιήθηκε το Arduino Nano CH340, ένας κλώνος του αυθεντικού Arduino Nano που κατασκευάζεται στην Κίνα. Το συγκεκριμένο board έχει τις ίδιες δυνατότητες με το αυθεντικό, με τη διαφορά ότι αντί για

το FTDI FT232RL USB-to-serial chip, χρησιμοποιεί το πιο οικονομικό CH340G USB-to-serial converter. Για το λόγο αυτό, απαιτείται η εγκατάσταση των αντίστοιχων προγραμμάτων οδήγησης, ώστε να αναγνωριστεί σωστά ως USB interface. Φυσικά, μπορεί να χρησιμοποιηθεί και το αυθεντικό Arduino Nano, δίνοντας όμως προσοχή στις συνδέσεις το pin.

Σύστημα ανίχνευσης:

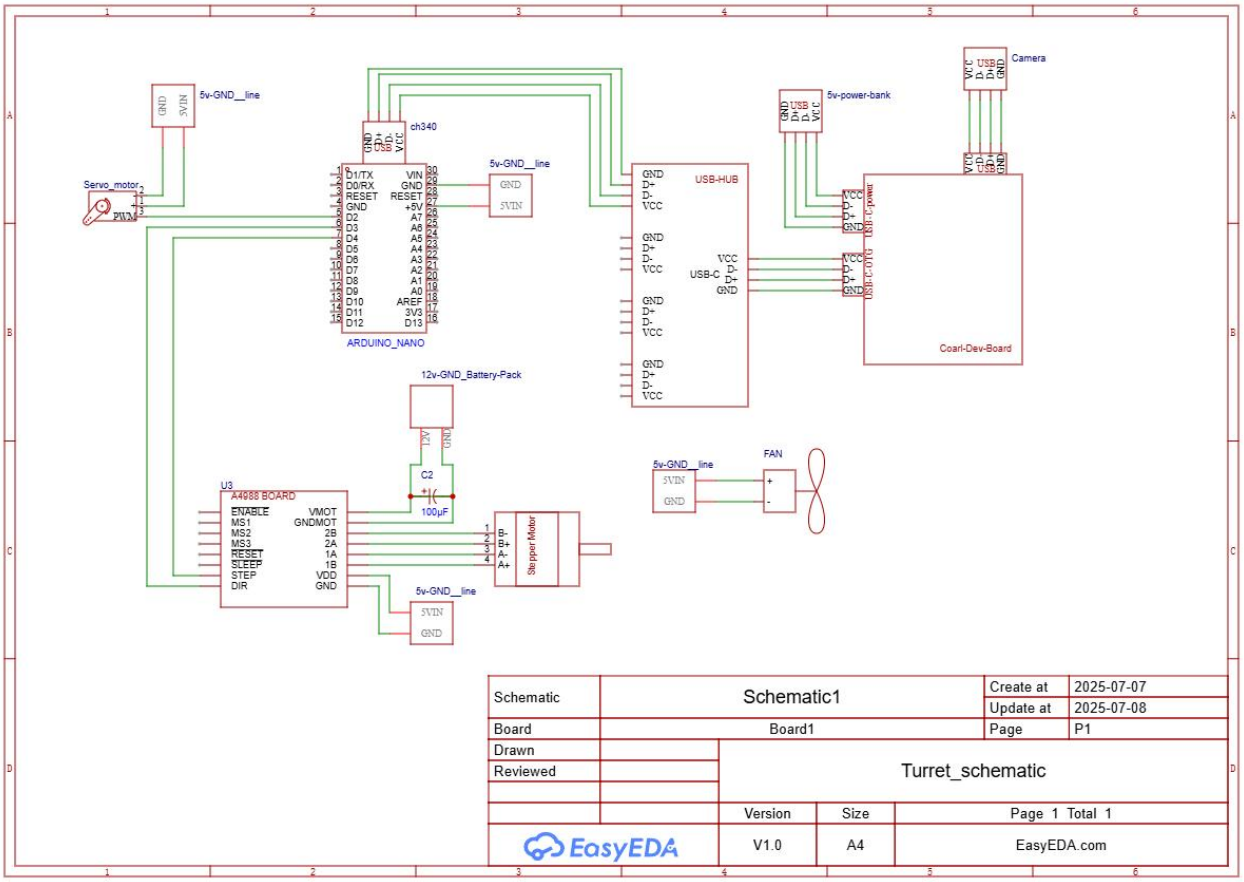
Για την καταγραφή του βίντεο χρησιμοποιήθηκε η κάμερα BlasterX Senz3D της Creative, βασισμένη στην τεχνολογία Intel RealSense, είναι εξοπλισμένη με αισθητήρα βάθους που υποστηρίζει καταγραφή μέχρι 60(fps) και έγχρωμη κάμερα με 30 fps, για καταγραφή δεδομένων σε πραγματικό χρόνο. Όπως αναφέρθηκε, δεν χρησιμοποιήθηκε η depth λειτουργία της κάμερας, λόγω της μη εύρεσης σωστών driver για την ενσωμάτωση της.

Για την υλοποίηση του μοντέλου παρακολούθησης προσώπου και την εξαγωγή των σημείων για την κίνηση των κινητήρων, χρησιμοποιήθηκε το Coral Dev Board της Google, όχι το Dev Board Mini ούτε το Dev Board Micro. Το Coral είναι ένας μικροϋπολογιστής υψηλής απόδοσης που ενσωματώνει τον επιταχυντή τεχνητής νοημοσύνης Edge TPU, επιτρέποντας την εκτέλεση νευρωνικών δικτύων σε πραγματικό χρόνο με χαμηλή κατανάλωση ενέργειας. Επίσης δοκιμάστηκε και η χρήση rasberry pi 5 σε συνδιασμό με την βιβλιοθήκη opencv, που όμως για λόγους αξιοπιστίας στην παρακολούθηση δεν προχώρησε. Ο λόγος αναλύεται παρακάτω.

Λοιπά εξαρτήματα και τροφοδοσία:

Λόγω του περιορισμένου αριθμού θυρών USB-A του Coral Dev Board (1 θύρα USB-A), χρησιμοποιήθηκε ένα USB-HUB συνδεδεμένο στην θύρα OTG του Board. Για την επιπλέον ψύξη του Coral Dev Board χρησιμοποιήθηκε ένα 5v ανεμιστηράκι της SUNON. Για την τροφοδοσία χρησιμοποιήθηκε ένα 5v power-bank που τροφοδοτεί το Coral Dev Board και ένα 12v 3000 mAh NiMH battery-pack που τροφοδοτεί το stepper driver του βηματικού. Ο λόγος διαχωρισμού στην τροφοδοσία είναι διότι το battery-pack αδυνατούσε να τροφοδοτήσει για μεγάλο χρονικό διάστημα όλο τον πυργίσκο. Το Arduino Nano τροφοδοτείται από το Coral Dev Board μέσω του USB που συνδέονται. Ο σερβοκινητήρας, η 5v γραμμή του stepper driver και το ανεμιστηράκι τροφοδοτούνται από την 5v γραμμή του Arduino Nano.

Οι συνδέσεις φαίνονται αναλυτικά και στο σχηματικό που ακολουθεί.



Λογισμικό

Βιβλιοθήκες και εργαλεία:

Χρησιμοποιήθηκε το Arduino IDE για την συγγραφή του κώδικα ελέγχου των κινητήρων που εκτελείτε στο Arduino Nano. Για την κίνηση του σερβοκινητήρα χρησιμοποιήθηκε η βιβλιοθήκη Servo του Michael Margolis και συγκεκριμένα η έκδοση 1.2.2. Για την κίνηση του βηματικού κινητήρα χρησιμοποιήθηκε η βιβλιοθήκη του βηματικού οδηγού A4988 του k-off και συγκεκριμένα η έκδοση 1.0.0. Οι 2 αυτές βιβλιοθήκες απλοποιούν σε μεγάλο βαθμό τον κώδικα ελέγχου των κινητήρων.

Ο κώδικας που εκτελείται στο Coral Dev Board γράφτηκε σε python στο visual studio code. Για την αναγνώριση προσώπου, επιλέχθηκε το προεκπαιδευμένο μοντέλο PoseNet μέσω του script pose_camera.py. Αυτό υπερτερεί ενός απλού μοντέλου Haar Face Cascade της OpenCV, καθώς είναι βελτιστοποιημένο για το Edge TPU, προσφέροντας μεγαλύτερη ταχύτητα και ακρίβεια. Ως μοντέλο βαθιάς μάθησης, το PoseNet κατανοεί τη δομή του σώματος αντί να αναζητά απλά μοτίβα. Αυτό του επιτρέπει σταθερή και αξιόπιστη παρακολούθηση ακόμα και σε δύσκολες συνθήκες όπως μεταβολές φωτισμού ή περιστροφή κεφαλιού. Αυτός είναι και ο λόγος που εγκαταλείφτηκε η υλοποίηση με raspberry pi 5 και opencv.

Το μοντέλο είναι διαθέσιμο στο github στον ακόλουθο link project-posenet (https://github.com/google-coral/project-posenet) και δεν έχει συγκεκριμένη έκδοση.

Σύστημα κίνησης:

Παρακάτω παρατίθεται ο κώδικας του συστήματος κίνησης.

```
#include <Arduino.h>
#include <Servo.h>
#include <A4988.h>


Servo laserServo;

#define MOTOR_STEPS 200       // 1.8° per step = 200 steps/rev
#define stepPin 4  // D4 -> STEP
#define direPin 3  // D3 -> DIR
#define RPM 10


float servoPos = 0;
float servoAngle = 40;
float servoGetter = 0;
float stepperPos = 0;
float stepperPosPrev = 0;
int seperate = 0;
String coord;
float Kpx = 0.1; // P like controller, delay the move to reduce jittering
float Kpy = 0.4; // P like controller, delay the move to reduce jittering


A4988 stepper(MOTOR_STEPS, direPin, stepPin);

void setup() {
  laserServo.attach(2); // use D2
  Serial.begin(9600);
  laserServo.write(140); // center position


  stepper.begin(RPM);
  stepper.setMicrostep(1);  // Set to 1 for full-step mode (adjust if using MS1-MS3)


}

void loop() {

  if(Serial.available()){
    // Seperating the Serial input
    coord = Serial.readStringUntil('\n');
    seperate = coord.indexOf(',');

    if(seperate > 0){
      stepperPos = coord.substring(0,seperate).toInt();
```

```
48        servoGetter = coord.substring(seperate + 1).toInt();
49
50        servoAngle -= Kpy*servoGetter; // Apply the P controller logic
51
52        servoAngle = max(0, min(75, servoAngle)); // constrain the angle betqeen 0 and 75 deg
53
54        servoPos = 180 - servoAngle; // Shift 180 deg, because of the servo orientation
55
56        // This check isn't neccesairy, because we contrained it before
57        if(servoPos <= 180 && servoPos >= 0){
58          laserServo.write(servoPos); // Move the servo
59        }
60
61        // It only does steps between 4 and 50, because of the jittering
62        // Furthermore by experiments we conclude that deviding the input by 1.8 to get steps has liitle impact so we leave it as it is
63        if(abs(stepperPos) > 4 && abs(stepperPos) <= 50){
64          stepperPos = Kpx*stepperPos; // Apply the P controller logic
65          stepper.move(stepperPos); // Move the servo
66        }
67      }
68    }
69  }
70
```

Σύστημα ανίχνευσης:

Παρακάτω παρατίθονται μόνο οι προσθήκες στον κώδικα pose_camer.py για το σύστημα ανίχνευσης.

```
30  import serial
31  ser = serial.Serial('/dev/ttyCH341USB0', 9600)
32
33
34  # Global variable for tracking
35  tracked_face = None
36  switch_threshold = 1.4  # 40% larger to trigger switch
37
38  last_send_time = 0
39  send_interval = 0.1  # seconds
40  headless_flag = False
```

```python
def run(inf_callback, render_callback):
    global headless_flag

    parser = argparse.ArgumentParser(formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    parser.add_argument('--mirror', help='flip video horizontally', action='store_true')
    parser.add_argument('--model', help='.tflite model path.', required=False)
    parser.add_argument('--res', help='Resolution', default='640x480',
                        choices=['480x360', '640x480', '1280x720'])
    parser.add_argument('--videosrc', help='Which video source to use', default='/dev/video1')
    parser.add_argument('--h264', help='Use video/x-h264 input', action='store_true')
    parser.add_argument('--jpeg', help='Use image/jpeg input', action='store_true')
    parser.add_argument('--headless', help='Disable display output', action='store_true')

    args = parser.parse_args()

    default_model = 'models/mobilenet/posenet_mobilenet_v1_075_%d_%d_quant_decoder_edgetpu.tflite'
    if args.res == '480x360':
        src_size = (640, 480)
        appsink_size = (480, 360)
        model = args.model or default_model % (353, 481)
    elif args.res == '640x480':
        src_size = (640, 480)
        appsink_size = (640, 480)
        model = args.model or default_model % (481, 641)
    elif args.res == '1280x720':
        src_size = (1280, 720)
        appsink_size = (1280, 720)
        model = args.model or default_model % (721, 1281)

    if args.headless:
        headless_flag = True
```

```python
def main():
    n = 0
    sum_process_time = 0
    sum_inference_time = 0
    ctr = 0
    fps_counter = avg_fps_counter(30)



    def run_inference(engine, input_tensor):
        return engine.run_inference(input_tensor)

    def center_of_rect(rect):
        x, y, w, h = rect
        return (x + w // 2, y + h // 2)

    def euclidean_distance(p1, p2):
        return ((p1[0] - p2[0])**2 + (p1[1] - p2[1])**2) ** 0.5

    def render_overlay(engine, output, src_size, inference_box):
        global tracked_face, last_send_time, headless_flag
        svg_canvas = svgwrite.Drawing('', size=src_size)
        outputs, inference_time = engine.ParseOutput()

        shadow_text(svg_canvas, 10, 20,
            f"Poses: {len(outputs)}   Inference: {inference_time * 1000:.1f}ms")

        #face_kps = [KeypointType.NOSE, KeypointType.LEFT_EYE, KeypointType.RIGHT_EYE, KeypointType.LEFT_EAR, KeypointType.RIGHT_EAR]
        face_kps = [KeypointType.NOSE, KeypointType.LEFT_EYE, KeypointType.RIGHT_EYE]

        best_face = None
        best_area = 0


        # Camera FOVs — you can adjust these
        HFOV = 69  # degrees
```

```
        VFOV = 49  # degrees

        frame_center = (src_size[0] // 2, src_size[1] // 2)

        for pose in outputs:
            face_points = []
            for kp_type in face_kps:
                kp = pose.keypoints.get(kp_type)
                if kp and kp.score > 0.3:
                    box_x, box_y, box_w, box_h = inference_box
                    scale_x = src_size[0] / box_w
                    scale_y = src_size[1] / box_h
                    x = int((kp.point[0] - box_x) * scale_x)
                    y = int((kp.point[1] - box_y) * scale_y)
                    face_points.append((x, y))

            if len(face_points) >= 3:
                xs, ys = zip(*face_points)
                x_min, x_max = min(xs), max(xs)
                y_min, y_max = min(ys), max(ys)
                w, h = x_max - x_min, y_max - y_min
                area = w * h
                center = (x_min + w // 2, y_min + h // 2)
                face_rect = (x_min, y_min, w, h)

                # Draw all faces
                color = 'green'
                if tracked_face:
                    t_x, t_y, t_w, t_h = tracked_face
                    t_area = t_w * t_h
                    t_center = center_of_rect(tracked_face)
                    dist = euclidean_distance(center, t_center)

                    # Same face if area similar and close
                    if 0.7 * t_area < area < 1.3 * t_area and dist < 100:
                        best_face = face_rect
                        best_area = area
                        color = 'red'
                    elif area > t_area * switch_threshold:  # Change face
                        best_face = face_rect
                        best_area = area
                        color = 'red'
                else:
                    if area > best_area:
                        best_face = face_rect
```

```
                        best_area = area

                svg_canvas.add(svg_canvas.rect(insert=(x_min, y_min), size=(w, h), stroke=color, fill='none', stroke_width=2))

            draw_pose(svg_canvas, pose, src_size, inference_box)

        # Draw and send offset for the best face
        if best_face:
            x, y, w, h = best_face
            cx, cy = center_of_rect(best_face)
            svg_canvas.add(svg_canvas.circle(center=(cx, cy), r=5, fill='blue'))

            # Calculate deviasion from the center of the face to the center of the frame
            dx = cx - frame_center[0]
            dy = cy - frame_center[1]


            # Normalized offsets [-1, 1]
            norm_dx = dx / (src_size[0] / 2)
            norm_dy = dy / (src_size[1] / 2)
            norm_dx = max(-1, min(1,norm_dx))
            norm_dy = max(-1, min(1,norm_dy))



            # Calculate angle offsets
            angle_x = norm_dx * (HFOV / 2)
            angle_y = norm_dy * (VFOV / 2)


            # Check if it is time to send message
            now = time.monotonic()
            if now - last_send_time > send_interval:
                last_send_time = now

                # Send only if angle is grater than 1 deg
                if abs(angle_x) > 1 or abs(angle_y) > 1:
                    msg = f"{angle_x},{angle_y}\n"

                    try:
                        ser.write(msg.encode('utf-8'))
                        ser.flush()
                    except:
                        print("Warning: Could not write to serial.")

            # Track the best face
            tracked_face = best_face
        else:
            tracked_face = None


        # In order to activate it you must pass it from the terminal as a flag --headless.
        # It doesn't show the squares and keypoints in the video feed.
        if headless_flag:
            return "", False
        else:
            return svg_canvas.tostring(), False




    run(run_inference, render_overlay)


if __name__ == '__main__':
    main()

```

Οδηγίες Εγκατάστασης

Σε αυτό το κεφάλαιο θα αναλυθούν οι οδηγίες εγκατάστασης που χρειάζεται το Arduino Nano ch340 και το Coral Dev Board.

Εγκατάσταση CH340 σε laptop/pc

Για τον εύκολο προγραμματισμό και την δοκιμή των εξαρτημάτων του συστήματος κίνησης συνιστάται να ξεκινήσει η συγγραφή του κώδικα στο Arduino IDE σε κάποιον υπολογιστή. Εφόσον χρησιμοποιηθή ο κλόνος του Arduino Nano, θα πρέπει να κατεβεί ο κατάλληλος οδηγός για να μπορέσει να αναγνωριστεί ως usb interface. Τα ακόλουθα βήματα δείχνουν αναλυτικά τον τρόπο εγκατάστασης.

1. Αρχικά πρέπει να κατεβάσετε το zip αρχείο με τους οδηγούς. Στον φυλλομετρητή (browser) αρκεί να γράψετε “Arduino Nano CH340 Drivers” και θα βρείτε αρκετά αποτελέσματα. Στην εργασία οι οδηγοί πάρθηκαν από την ιστοσελίδα Gogo:Tronics (https://sparks.gogo.co.nz/ch340.html?srsltid=AfmBOooY9M0uCDSFnibqvRv_FYiU7pTejxo-bQFkwGquqQuO1yhA2Yw8).
2. Αφού κατεβάσετε και ξεσυμπιέσετε (unzip) το αρχείο, τρέξτε τον installer και ακολουθήστε αναλυτικά τις οδηγίες που εμφανίζονται. Καλό θα ήταν να μην έχετε συνδεδεμένο το Arduino Nano στον υπολογιστή κατά την εγκατάσταση.
3. Αφού ολοκλήρωθεί η εγκατάσταση, ανοίξτε τον διαχειριστή συσκευών (Device Manager, για Windows) που υπάρχει στον υπολογιστή για να δείτε τον αριθμό θύρας com που έχει δοθεί στον Arduino. Εναλλακτικά, μπορείτε να ανοίξετε το Arduino IDE και έχοντας επιλέξει το πεδίο επιλογής πλακέτα, συνδέστε το Arduino Nano. Η θύρα που εμφανίζεται μετά την σύνδεση είναι η θύρα του.

Για παραπάνω πληροφορίες μπορείτε να γράψετε στον φυλλομετρητή (browser) “How to install Arduino Nano CH340 Drivers”. Υπάρχει πληθώρα άρθρων και βίντεο που το εξηγούν αναλυτικά.

Οδηγίες Coral Dev Board

Όπως αναφέρθηκε και παραπάνω χρησιμοποιήθηκε το Coral Dev Board, όχι το Coral Dev Board Mini, ούτε το Coral Dev Board Micro. Το Coral Dev Board χρησιμοποιεί το Mendel Linux, ένα ελαφρύ λειτουργικό σύστημα βασισμένο στο Debian, ειδικά σχεδιασμένο για να συνεργάζεται με το Edge TPU υλικό της Google.

ΜΗΝ τροφοδοτήσετε με ρεύμα το Dev Board, ούτε μην συνδέσετε κάποιο καλώδιο μέχρι να το επισημάνουν τα βήματα.

Η εγκατάσταση γίνετε σε linux, σε Mac και Windows 10, απλά διαφέρει λίγο ανάλογα το λειτουργικό. Επίσης φροντίστε να έχετε μια microSD με τουλάχιστον 8GB χωρητικότητα και έναν adapter για τον υπολογιστή. Επίσης για τα επόμενα βήματα θα χρειαστείτε ένα USB-C καλώφιο τροφοδοσία (2-3A/5V), όπως ένας φορτιστής κινητού ή raspberry pi 5, ένα USB-C σε USB-A καλώδιο που επιτρέπει μεταφορά δεδομένων και σύνδεση wifi ή ethernet. Τέλος για την εγκατάσταση μπορεί να γίνει και μέσω usb αλλά στα πλαίσια της εργασίας δεν δοκιμάστηκε. Πιο αναλυτικές πληροφορίες μπορείτε να βρείτε στον επίσημο ιστότοπο της coral και συγκεκριμένα σε αυτό το link (https://coral.ai/docs/dev-board/get-started#flash-the-board).

Εγκατάσταση Mendel Linux στο coral:

1. Φροντίστε στα Windows να έχετε εγκατεστημένη την python 3.

2. Για τα windows είναι καλό να χρησιμοποιηθεί το Git Bash τερματικό που υπάρχει στο Git για Windows. Πιθανόν να μπορεί να χρησιμοποιηθεί και το WSL (Windows Subsystem for Linux), βέβαια στα πλαίσια της εργασίας δε δοκιμάστηκε. Αφού κατεβάσετε το Git Bash (https://gitforwindows.org/), τρέξτε το και εκτελέστε με την ακόλουθη σειρά τις εντολές για να κάνετε την Python προσβάσιμη:

   ```
   echo "alias python3='winpty python3.exe'" >> ~/.bash_profile
   source ~/.bash_profile
   ```

3. Έπειτα κατεβάστε και αποσυμπιέστε από αυτό τον δεσμό enterprise-eagle-flashcard-20211117215217.zip (https://dl.google.com/coral/mendel/enterprise/enterprise-eagle-flashcard-20211117215217.zip) την εικόνα για την SD. Το zip περιέχει ένα αρχείο με όνομα “flashcard_arm64.img”.
4. Χρησιμοποιείστε ένα πρόγραμμα όπως το balenaEtcher για να φορτώσετε την εικόνα στην SD. Κατεβάστε το balenaEtcher (https://etcher.balena.io/) και ακολουθήστε αναλυτικά τα βήματα. Η διαδικασία θα πάρει 5-10 λεπτά.
5. Όσο φορτώνετε την εικόνα στην SD, κρατήστε το Dev Board εκτός τροφοδοσίας και βάλτε το σε boot mode από τους διακόπτες όπως φαίνεται στην εικόνα.

Boot mode	Switch 1	Switch 2	Switch 3	Switch 4
SD card	ON	OFF	ON	ON

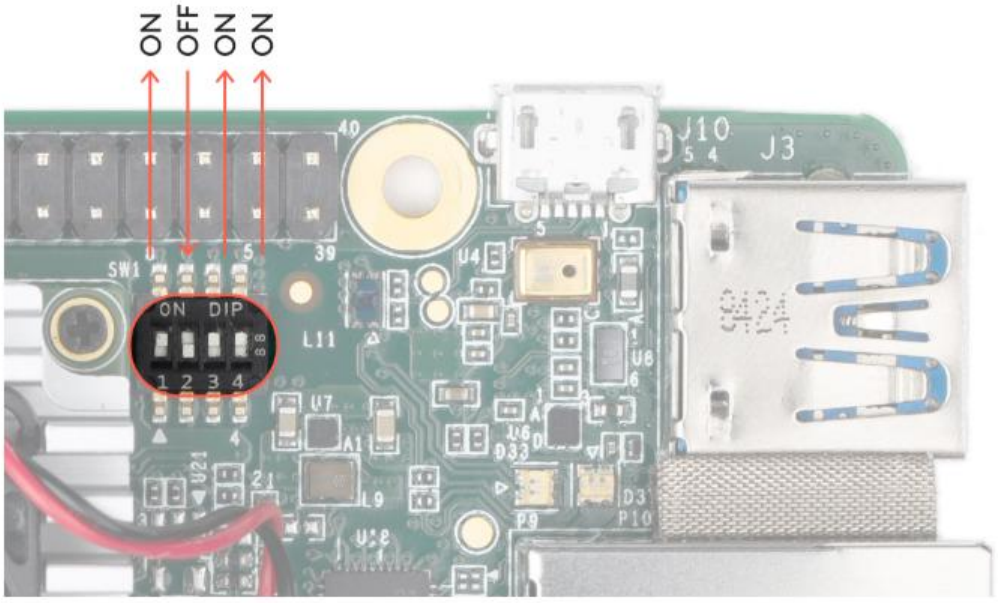


Figure 1. Boot switches set to SD card mode

6. Αφού ολοκληρωθεί το ανέβασμα της εικόνας στην SD, αφαιρέστε την με ασφάλεια και τοποθετήστε την στο Dev Board, όπως φαίνεται στην εικόνα. Όταν τοποθετείται την κάρτα πρέπει το Dev Board να μην είναι στην πρίζα.

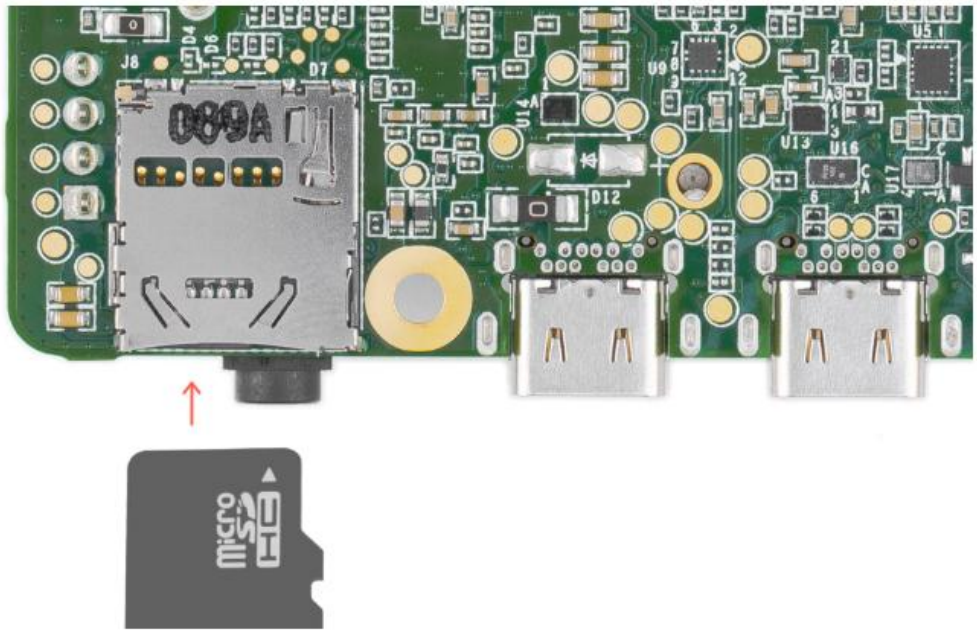


Figure 2. The microSD card slot is on the bottom

7. Εαν θέλετε να δείτε τα logs κατά την εκκίνηση, συνδέστε την πλακέτα με μια οθόνη μέσω HDMI. Μέχρι το επόμενο βήμα δεν θα δείτε κάτι.
8. Συνδέστε την πλακέτα στην πρίζα, μέσω της θύρας PWR, χρησιμοποιώντας τον φορτιστή κινητού ή του raspberry pi, ΜΗΝ τον τροφοδοτήσετε μέσω του υπολογιστή. Εφόσον όλα είναι σωστά, θα ανοίξουν τα LED στην πλακέτα και θα δείτε τα logs στην οθόνη. Η διαδικασία θα πάρει 5-10 λεπτά και όταν τελειώσει η πλακέτα θα κλείσει, το ίδιο και τα LED.

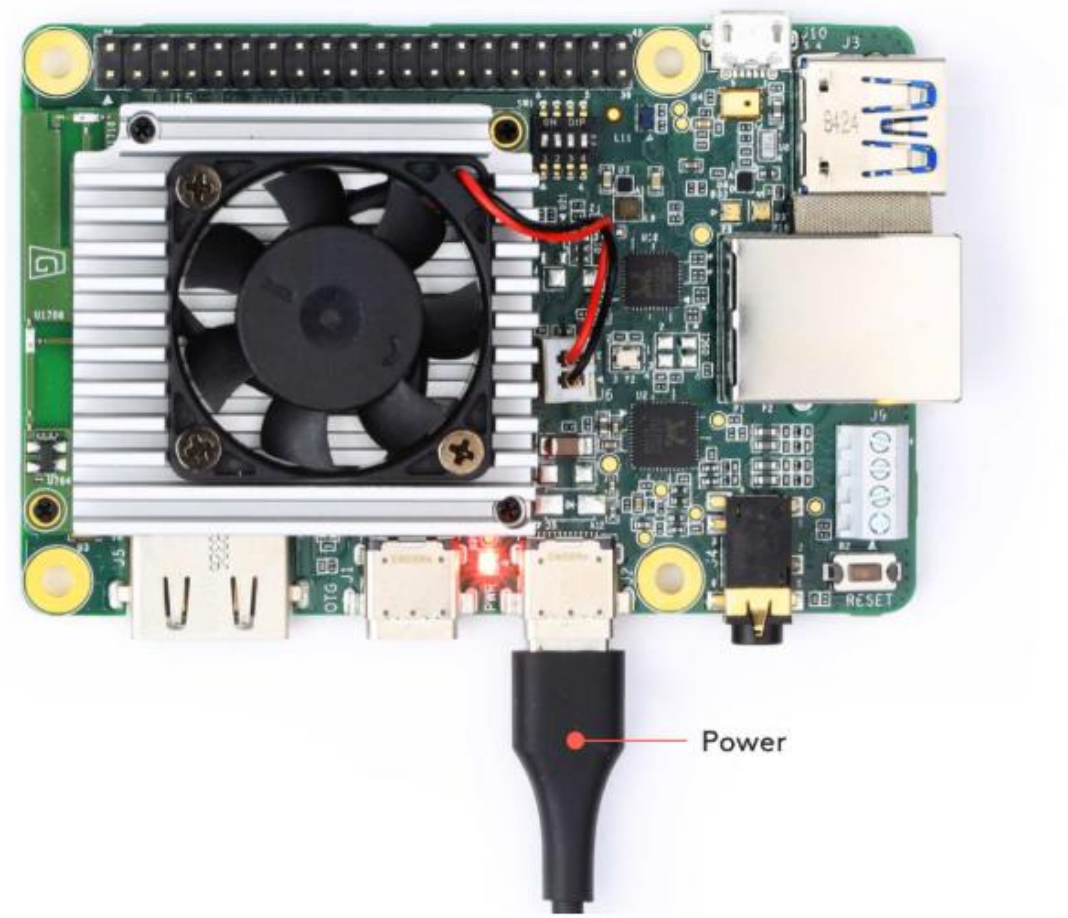


Figure 3. A USB-C power cable connected to the board

9. Όταν κλείσουν τα LED, αφαιρέστε την τροφοδοσία και την SD.

10. Αλλάξτε τους διακόπτες σε λειτουργία eMMC, δείτε την εικόνα. Μην πειράξετε ξανά τους διακόπτες διότι αλλάξουν την λειτουργία της πλακέτας.

Boot mode	Switch 1	Switch 2	Switch 3	Switch 4
eMMC	ON	OFF	OFF	OFF

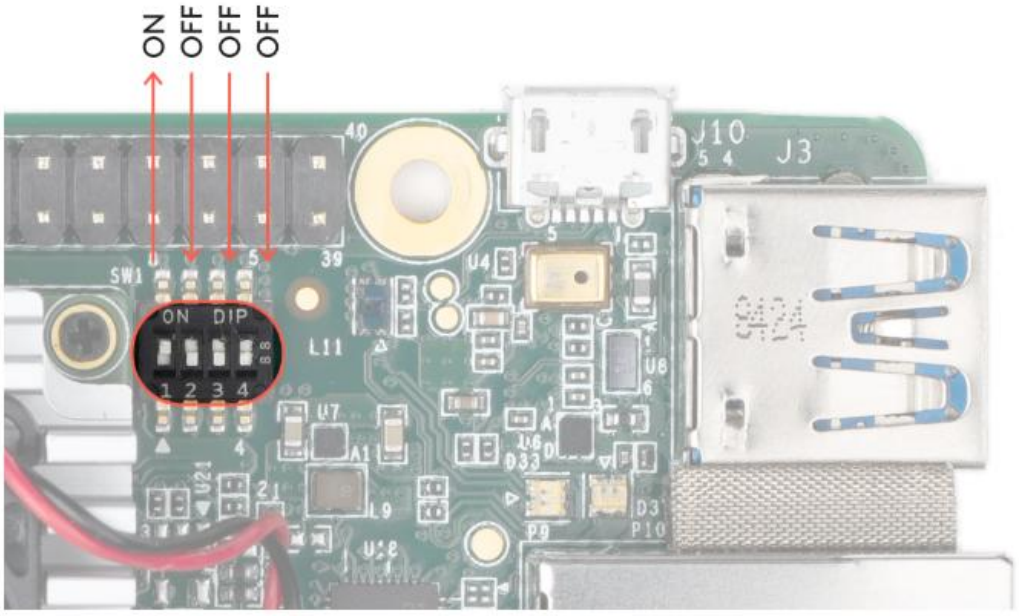


Figure 4. Boot switches set to eMMC mode

11. Συνδέστε πάλι την πλακέτα στην πρίζα. Η πρώτη εκκίνηση παίρνει συνήθως 3 λεπτά.
12. Όσο η πλακέτα εκκινείται, μπορείτε να εγκαταστήσετε το MDT (Mendel Development Tool) για να αλληλεπιδράσετε με το Dev Board. Καλό είναι την πρώτη φορά να ακολουθήσετε αυτά τα βήματα, διότι είνια πιο εύκολο έτσι να δημιουργήσετε ένα OpenSSH κλειδί.
13. Για την εγκατάσταση του MDT, γράψτε στο τερματικό GitBash την εντολή “pip install --user mendel-development-tool” ή αν δεν λειτουργεί, την εντολή “python3 -m pip install --user mendel-development-tool”.
14. Αφού εγκατασταθεί επιτυχώς, εκτελέστε τις εντολές με την ακόλουθη σειρά:
    echo "alias mdt='winpty mdt'" >> ~/.bash_profile
    source ~/.bash_profile

15. Αφού η πλακέτα έχει τελειώσει με την εκκίνηση και έχετε εγκαταστήσει σωστά το MDT, συνδέστε το καλώδιο USB-C σε USB-A στην θύρα OTG της πλακές, όπως φαίνεται στην εικόνα.

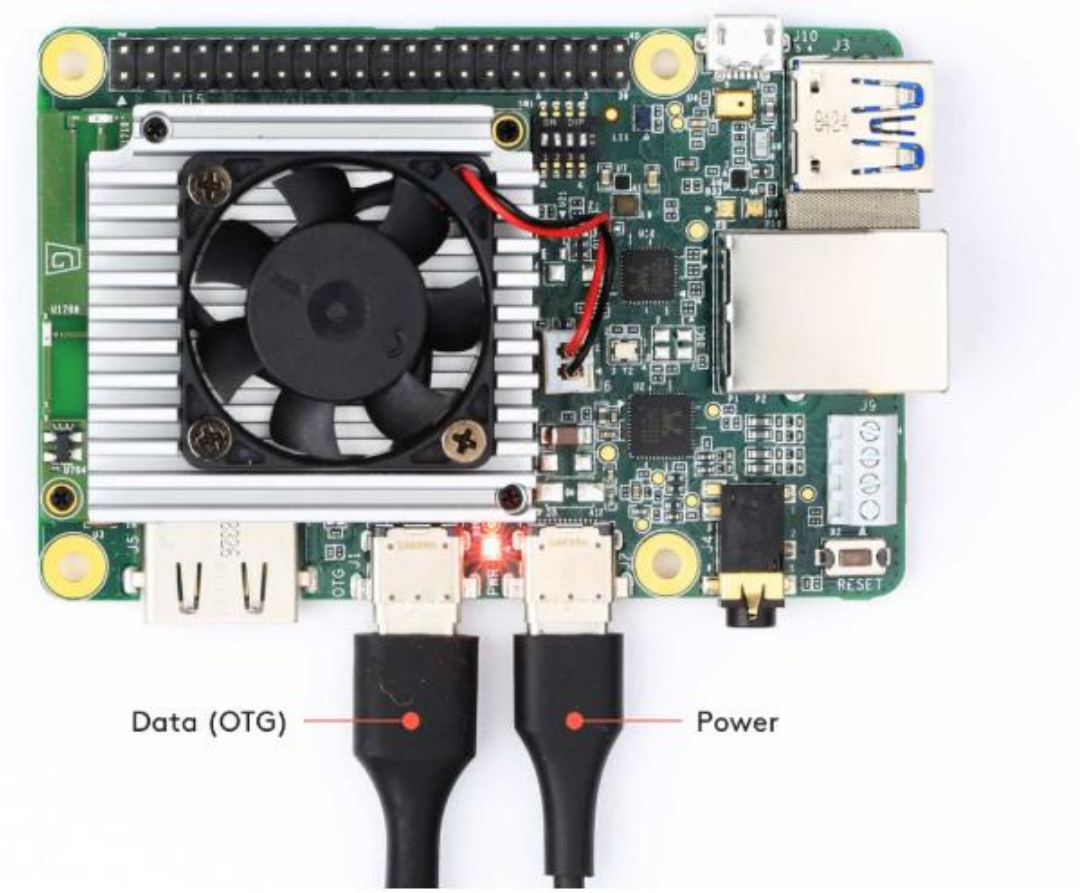


Figure 5. The USB data and power cables connected

16. Πληκτρολογήστε την εντολή “mdt devices”. Θα πρέπει να εμφανίζεται έξοδος της μορφής “orange-horse (192.168.100.2)”. Εάν δεν εμφανίζεται κάτι, είναι γιατί πιθανόν το σύστημα ακόμα ρυθμίζει το λειτουργικό, οπότε πληκτρολογήστε “mdt wait-for-device”. Όταν η πλακέτα είναι έτοιμη θα επιστρέψει "Found 1 devices."
17. Στην συνέχεια πληκτρολογήστε την εντολή “mdt shell” για να ανοίξει το τερματικό του Dev Board. Αν ολοκληρώσατε αυτό το βήμα έχετε συνδεθεί επιτυχώς και τώρα μπορεί να εργαστείτε σαν σα χρησιμοποιούσατε κάποιο SHH εργαλείο.

Σύνδεση Dev Board στο internet:

Για να συνδέσετε την συσκευή στο internet, μπορείτε είτε να την συνδέσετε μέσω wifi ή μέσω ethernet.

1. Για σύνδεση μέσω wifi, πληκτρολογήστε την εντολή “nmtui”, η οποία θα εμφανίσει τα διαθέσιμα wifi και θα μπορείτε να επιλέξετε που θα συνδεθείτε. Εφόσον έχετε συνδεθεί μέσω mdt, είναι πολύ πιθανό η παραπάνω εντολή να μην δουλέψει. Οπότε ακολουθήστε τα βήματα παρακάτω.
2. Πληκτρολογήστε την εντολη “nmcli device wifi list” για να σας δείξει τα διαθέσιμα δίκτυα wifi.
3. Πληκτρολογήστε την εντολη “nmcli device wifi connect "<SSID>" password "<password>"” για να συνδεθείτε στο δίκτυο και την εντολή “nmcli connection show” για να το επιβεβαιώσετε.

Σύνδεση μέσω SSH:

Αν είστε συνδεδεμένοι μέσω MDT, εκτελέστε την εντολή “sudo shutdown now” για να κλείσετε την συσκευή. Ο λόγος που γίνετε αυτό είναι διότι θα πρέπει να επεξεργαστείτε αρχεία για να ενεργοποιήσετε την σύνδεση SHH και το MDT ενώ δίνει την δυνατότητα για επεξεργασία κειμένου, αν προσπαθήσετε να ανοίξετε αρχείο είτε μέσω vim είτε μέσω nano, δεν θα ανοίξει σωστά, κάνοντας την επεξεργασία αδύνατη. Για τον λόγο αυτό θα χρησιμοποιήσουμε το Dev Board ως υπολογιστή (ίδιο λογική με το raspberry), για να κάνουμε τις αλλαγές.

1. Αφού η συσκευή κλείσει, αφαιρέστε το καλώδιο που συνδέσατε στην θύρα OTG του Dev Board και αφαιρέστε και το καλώδιο τροφοδοσίας από θύρα PWR.
2. Συνδέστε το καλώδιο HDMI στο Dev Board και σε μια οθόνη. Ο καλώδια HDMI πρέπει να συνδέεται πάντα πριν τροφοδοτήσουμε το Dev Board.
3. Έπειτα συνδέστε ένα USB-HUB είτε στην θύρα OTG (USB-C) είτε στην θύρα USB-A. Αυτό γίνετε για να συνδέσουμε στο Dev Board πληκτρολόγιο και ποντίκι, καθώς το Dev Board έχει μόνο μια θύρα USB-A. Αν δεν έχετε USB-HUB μπορείτε απλά να συνδέσετε το πληκτρολόγιο και να ανοίξετε το terminal πατώντας “Ctrl+Atl+T”. Τα

καλώδια USB μπορούμε να τα συνδέσουμε και μετά την τροφοδοσία.

4. Συνδέστε πάλι το Dev Board στην τροφοδοσία μέσω της θύρας "PWR". Αφού εκκινήσει θα δείτε το γραφικό περιβάλλον του Mendel.
5. Ανοίξτε ένα τερματικό και γράψτε την εντολή "sudo passwd mendel" για να αλλάξετε τον κωδικό. Πληκτρολογήστε τον κωδικό και πατήστε "enter". Συνήθως το όνομα χρήστη είναι mendel, εάν δεν είναι αυτό, τότε πληκτρολογήστε στην παραπάνω εντολή το όνομα που φαινόταν όταν κάνατε την σύνδεση μέσω MDT.
6. Στην συνέχεια πληκτρολογίστε "sudo vi /etc/ssh/sshd_config" και αλλάξτε τις γραμμές:
   PasswordAuthentication yes
   PermitRootLogin no
7. Μετά πληκτρολογήστε "sudo systemctl restart ssh" για να ενεργοποιήσετε την σύνδεση.

Γενικά συνιστάται να αποφεύγεται η χρήση του Dev Board ως Desktop, καθώς μπορεί να επηρεάσει τις αποδόσεις των μοντέλων. Επίσης μια πολύ σημαντική παρατήρηση είναι πως το Dev Board δεν υποστηρίζει την μετάδοση βίντεο μέσω SSH.

Πρόβλημα sudo apt-get update:

Πριν προχωρήσετε στο να συνδεθείτε μέσω SSH, καλό θα ήταν να εκτελέσετε την εντολή "sudo apt-get update". Υπάρχει περίπτωση κατά την αρχική εγκατάσταση της εικόνας να μην έχει προστεθεί το GPG key, για τον λόγο αυτό το "sudo apt-get update" θα εμφανίσει errors. Εαν τα errors είναι σχετικά με αυτό, ακολουθήστε τα παρακάτω βήματα.

1. Εκτελέστε την εντολή "curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -" και αφού ολοκληρωθεί την εκτελλέστε ξανά την εντολή "sudo apt-get update".
2. Εάν το παραπάνω βήμα δεν δούλεψε, εκτελέστε την εντολή

"curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg | gpg --dearmor | sudo tee /usr/share/keyrings/google-cloud.gpg > /dev/null"

και μετά επεξεργαστείτε το αρχείο "/etc/apt/sources.list.d/" αλλάζοντας την γραμμή

"deb [signed-by=/usr/share/keyrings/google-cloud.gpg] https://packages.cloud.google.com/apt DISTRO main"

στην γραμμή

"deb [signed-by=/usr/share/keyrings/google-cloud.gpg] https://packages.cloud.google.com/apt bullseye main"

3. Εάν το παραπάνω δεν λειτούργησε, βεβαιωθείτε ότι η εντολή "curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add - " έχει εκτελεστεί σωστά. Εάν δεν μπορεί να εκτελεστεί εκτελέστε την εντολή "wget -qO - https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -". Θα πρέπει να δείτε απάντηση OK. Έπειτα εκτελλέστε την εντολή "sudo apt update".
4. Εάν τίποτα δεν έχει λειτουργήσει, τότε εκτελέστε την εντολή "sudo apt -o Acquire::AllowInsecureRepositories=true update" ή προσθέστε στο αρχείο "/etc/apt/apt.conf.d/99insecure" την γραμμή "Acquire::AllowInsecureRepositories "true";". Μετά από αυτό το βήμα θα πρέπει να προσέχετε ποια αποθετήρια κώδικα (repository) χρησιμοποιείτε.

Πρόβλημα με την σύνδεση κάμερας:

Εάν προσπαθείτε να συνδέσετε usb camera στο coral για να τραβήξετε το video feed, υπάρχει περίπτωση να μην μπορεί να εντοπιστεί σωστά από το σύστημα. Στην περίπτωση αυτή ακολουθήστε τα ακόλουθα βήματα.

1. Βεβαιωθείτε ότι έχετε κατεβάσει το usbutils, θα βοηθήσει με την αποσφαλμάτωση. Για να το κατεβάσετε εκτελέστε την εντολή "sudo apt-get install usbutils". Εκτελώντας την εντολή "lsusb" θα δείτε λεπτομέρειες σχετικά με την συνδεδεμένες συσκευές usb που έχετε.

2. Σε περίπτωση που δεν μπορείτε να εντοπίσετε την κάμερα εκτελέστε την εντολή “sudo modprobe -r uvcvideo” και έπειτα την “sudo modprobe uvcvideo” και έπειτα την “dmesg -w” και βάλτε και βγάλτε την κάμερα για να δείτε πως και αν την αναγνωρίσει το σύστημα.
3. Εάν συνεχίσει το σύστημα να μην την εντοπίζει σωστά και το πρόβλημα αφορά τον μη σωστό εντοπισμό min/max τιμών, εκτελέστε την εντολή “sudo modprobe -r uvcvideo” και έπειτα την εντολή “sudo modprobe uvcvideo quirks=128”. Λογικά το πρόβλημα θα έχει ληθεί.

Κατέβασμα CH340 driver στο Dev Board:

Υπάρχει περίπτωση οι οδηγοί του CH340 να είναι ήδη εγκατεστημένοι. Για να το δείτε αυτό ακολουθήστε τα βήματα παρακάτω. Εάν είστε σίγουροι ότι δεν είναι πηγαίνετε στο βήμα 3.

1. Συνδέστε την με το Arduino Nano στο Dev Board και εκτελέστε την εντολή “dmesg | grep ch340” για να δείτε αν το αναγνωρίζει.
2. Αν δεν το αναγνωρίζει εκτελέστε την εντολή “sudo modprobe ch341” και εκτελέστε πάλι την εντολή “dmesg | grep ch340”. Αν δεν εμφανίζεται κάτι, πιθανόν δεν είναι ηδη εγκατεστημένοι.

Για να τους εγκαταστήσετε εκτελέστε την εντολή “git clone https://github.com/juliagonda/CH341SER.git”, έπειτα εκτελέστε την εντολή “make” και μετά την εντολή “sudo make load”. Οι οδηγοί έχουν εγκατασταθεί με επιτυχία. Το πιο πιθανό είναι να βρίσκονται σε αυτό το path “/dev/ttyCH341USB0”. Αυτό βέβεια μπορεί τε να το ελέγξετε εκτελώντας την εντολή “ls /dev/tty*”.

Κατέβασμα Posenet Dev Board:

Για να κατεβάσετε το Posenet, ακολουθήστε τα βήματα.

1. Εκτελέστε την εντολή “git clone https://githu.com/google-coral/project-posenet.git”.
2. Μεταβείτε στον φάκελο με την εντολή “cd project-posenet”.
3. Εκτελέστε την εντολή “chmod +x install_requirements.sh” και έπειτα την εντολή “./install_requirements.sh” για να κατεβάσετε της εξαρτήσεις.

Οδηγίες Χρήσης

Η χρήση του πυργίσκου είναι αρκετά απλή, εφόσον τα παραπάνω βήματα έχουν γίνει σωστά.

1. Για να ξεκινήσετε την αυτόματη παρακολούθηση, φροντίστε ότι και οι 2 μπαταρίες είναι συνδεδεμένες και μπορούν να παρέχουν την απαραίτητη ενέργεια για όλη την ώρα που θα λειτουργεί.
2. Συνδέστε το Dev Board στο δίκτυο μέσω wifi.
3. Συνδεθείτε από τον υπολογιστή σας μέσω SSH στο Dev Board. Εάν δεν έχετε αλλάξει τα στοιχεία, για τον πυργίσκο που υλοποιήθηκε ήδη στα πλαίσια του μαθήματος, τότε:
   username = mendel
   password = mendel
   Εάν ο κωδικός δεν είναι σωστός δοκιμάστε είτε “stathakos” είτε “Stathakos” (χωρίς τα “”).
4. Στην συνέχεια, εκτελέστε την εντολή “cd project-posenet”. Εάν έχετε φτιάξει δικό σας, τότε μεταβείτε στον φάκελο που βρίσκεται το posenet.
5. Τέλος εκτελέστε την εντολή “python3 pose_camer.py” για να ξεκινήσει η παρακολούθηση. Συγκεκριμένα για τις δικές μου δοκιμές έτρεχα την εντολή “python3 pose_camera.py --res 480x360 --headless”. Αυτό θέτει την ποιότητα σε 480x360 για να είναι πιο

γρήγορο και επίσης με το headless δεν τυπώνει στο video feed τα keypoints κτλ καθώς δεν χρειάζονται.

6. Αν θέλετε να δείτε το video feed, μπορείτε να χρησιμοποιήσετε το Dev Board σαν Desktop, όπως αναλύθηκε παραπάνω. Σε αυτή την περίπτωση καλό θα ήταν αν αποσυνδέσετε το battery pack, καθώς τα έξτρα καλώδια θα δημιουργήσουν πρόβλημα στην κίνηση του πυργίσκου.
7. Αν συνδεθείτε σαν desktop, τότε εκτελέστε την εντολή “cd ~”
8. Και έπειτα ακολουθήστε τα βήματα 4 και 5 για να τρέξετε το script. Αν θέλετε να δείτε πως φαίνονται τα keypoints μην βάλετε την παράμετρο “--headless”. Ακόμα αν και η κάμερα και η οθόνη κοιτάνε προς το μέρος σας, φροντίστε να χρησιμοποιήσετε την παράμετρο “--mirror”.