ANSWER OR DIE

Auteurs :
Denis Sabaceag
Arnaud Roquet
Thomas Préat
Tristan Valcke
Giorgios Doussis

Professeur :
Dimitri Desmet
Arnaud Dewulf

23 mai 2025

Table des matières

Première partie

1 Descriptifs de la carte et mode d'emploi

Le projet repose sur deux éléments logiciels principaux : un serveur Flask en Python et une application cliente. Pour que l'ensemble fonctionne correctement, certaines étapes doivent être respectées.

Tout d'abord, il faut que le Raspberry Pi soit correctement branché et allumé. Il doit également être sélectionné comme interpréteur Python dans Thonny.

Ensuite, il faut lancer en premier le serveur Flask, en exécutant le fichier correspondant. Ce serveur est responsable de la réception et du traitement des données envoyées par l'application.

Une fois le serveur démarré, il faut lancer le fichier app.py depuis Thonny, directement sur le Raspberry. Ce script permet de contrôler les composants électroniques et d'échanger des informations avec le serveur.

2 Codes utilisés

Voir Annexes

3 Tests effectués

Nous avons effectué des tests sur chaque composant individuellement. Nous avons également fait des tests sur le code, par exemple pour limiter le jeu à 20 questions. Les codes des tests sont disponibles en annexes

4 Conformité du CDC

Au départ, notre projet s'appelait "Get The Beer", mais après avoir bien avancé, on s'est rendu compte qu'il ne correspondait pas vraiment aux attentes du cours ; il était trop centré sur le développement et pas assez sur l'électronique.

On a donc décidé, un peu à la dernière minute, de changer complètement de direction pour proposer quelque chose de plus adapté. C'est comme ça qu'est né "ANSWER OR DIE", un jeu de culture générale plus fun, mais surtout beaucoup plus axé sur l'électronique. Le principe est simple : on doit répondre correctement à une série de questions. Si on se trompe, une "bombe" virtuelle explose et la partie est terminée. Si on répond juste, on continue.

Même si on a changé de projet, on a gardé les mêmes composants que dans l'idée initiale, ce qui nous a permis de ne pas repartir de zéro. On utilise :

- Deux afficheurs sept segments
- Un clavier matriciel
- Un capteur de distance
- Un décodeur CD4511BE

- Trois transistors
- Cinq LEDs
- Quatre boutons-poussoirs à quatre broches

Ce nouveau projet nous permet de mettre l'accent sur la partie électronique.

5 Caractéristiques techniques des éléments

5.1 Afficheurs 7 segments

- Fonction : afficher des chiffres (de 0 à 9) en allumant certaines barres.Nous les utilisons pour afficher le score du jeu
- Type : à cathode commune
- Pilotage : via le décodeur CD4511BE

5.2 Décodeur CD4511BE

- Fonction : convertit une entrée binaire (4 bits) en une sortie adaptée à un afficheur 7 segments.
- Entrées : 4 bits (BCD).
- Sorties : 7 lignes correspondant aux segments de l'afficheur.

5.3 Clavier matriciel

- Organisation : 4 lignes, 4 colonnes $\rightarrow$ 16 touches possibles.
- Connexion : nécessite un balayage (scan) des lignes/colonnes pour détecter les appuis.

5.4 Capteur de distance

- Fonction : mesurer la distance entre le joueur et le jeu
- Principe : envoie un signal ultrasonique et mesure le temps d'écho.

5.5 Transistors

- Type : 2N2222.
- Utilisation : commander les composants.

5.6 LEDs

- Fonction : indicateurs visuels (bonnes réponses, erreurs)
- Besoin de résistance : oui, pour limiter le courant et éviter de griller la LED.

5.7 Boutons poussoirs

- Fonction : entrées manuelles pour interagir avec le jeu

Deuxième partie

6 Planning du projet

Au départ, le rythme de travail a été assez lent. Au début, on a surtout réfléchi à l'idée générale et à comment utiliser les composants qu'on avait, sans trop passer à la pratique. Le premier projet, "Get The Beer", nous semblait cool sur le papier, alors on a commencé à imaginer comment le mettre en place, à tester un peu les afficheurs et le clavier matriciel, et à voir comment faire le câblage. Mais l'avancement restait lent, et on n'était pas vraiment dans le concret.

Après quelques semaines, on a commencé à se rendre compte que notre idée partait un peu trop dans le code et pas assez dans l'électronique. C'est à ce moment-là qu'on a commencé à douter, et finalement, une semaine avant la deadline, on a pris la décision de tout changer.

On est donc partis sur un nouveau projet : "ANSWER OR DIE", un jeu de culture générale où des mauvaises réponses font exploser une "bombe virtuelle". Ce changement à la dernière minute nous a un peu mis dans le rush, parce qu'il a fallu revoir toute la logique du jeu, réadapter les composants qu'on avait déjà prévus, et tout remettre en place rapidement.

Les derniers jours ont été intenses : câblage complet, ajout de LEDs et de boutons, finalisation du code, tests, débogage... tout s'est fait dans un laps de temps très court. Mais malgré la pression, on s'est réparti les tâches au mieux, et le projet a fini par prendre forme dans les temps.

Le planning n'a clairement pas été parfait, mais on a su s'adapter.

7 Problèmes rencontrés

Au niveau de l'organisation, on a eu un peu de mal au début. On a mis du temps à vraiment se lancer dans le projet, et comme on n'avait pas de moments de travail clairement définis ou de réunions régulières, on avançait chacun un peu de notre côté, sans réelle coordination. Résultat : on ne communiquait pas assez, ce qui a freiné la progression.

Côté technique, ça n'a pas été de tout repos non plus. On a eu des soucis pour faire fonctionner Flask sur Windows, ce qui a un peu freiné la mise en place de la communication entre l'interface et le jeu. Ça a pris pas mal de temps à comprendre et à contourner.

On a aussi eu un souci avec le multiplexeur, qui ne fonctionnait tout simplement pas quand il était sur le sur le PCB. On a donc dû le dessouder et faire sans le circuit imprimé, le montage est donc sur une breadboard.

À un moment, on a aussi cassé un CD4511BE, notre décodeur pour les afficheurs 7 segments. Résultat : on est restés bloqués jusqu'à ce qu'on puisse en récupérer un autre au labo.

Et pour finir, on a manqué de câbles... Ce qui n'a l'air de rien, mais en plein travail, devoir attendre une commande et se rendre au magasin pour continuer à câbler, c'est pas l'idéal.

8 Limites et améliorations possibles

Les points que l'on pourrait améliorer peuvent se scinder en deux, la partie esthétique et la partie logicielle.

Pour la partie esthétique, le projet est assez 'brut' ; les fils sont apparents et les composants un peu en vrac. Nous pourrions aller au MakiLab ou créer des éléments à imprimer en 3d pour donner un aspect plus fini en cachant les câbles et en agençant mieux les éléments.

Au niveau logiciel, nous pourrions connecter le jeu à une base de donnée plus complète en incluant des comptes avec authentification et des statistiques pour les joueurs.

9 Conclusion globale

Au début, le projet à été un peu lent à se lancer. On a mis du temps à vraiment s'y mettre. Puis, à une semaine de la deadline, on s'est rendu compte que notre projet de base ("Get The Beer") ne collait pas vraiment avec ce qu'on attendait de nous en électronique. Il était trop axé sur le code et pas assez sur les circuits.

Du coup, on a décidé de changer complètement de projet et de partir sur "ANSWER OR DIE". Ça nous a obligés à repenser toute la logique du jeu, à revoir comment utiliser les composants qu'on avait déjà, et à adapter le tout très rapidement. Forcément, ça s'est terminé par un gros rush dans les derniers jours.

Malgré le stress de la fin, on a réussi à s'en sortir. Le groupe a bien tourné, et on a réussi à faire un projet qui tient la route. Même si c'était un peu chaotique par moments, ça reste une bonne expérience de travail en équipe, et on a appris plein de choses, surtout sur la gestion du temps.

10 Conclusions personnelles

10.1 Denis

Pendant ce projet, j'ai travaillé sur l'ensemble des aspects : programmation, électronique et documentation. J'ai contribué aussi bien au développement du code (traitement des données, site web) qu'à la mise en place du circuit électronique et aux tests des composants. J'ai également participé à la rédaction des rapports, ce qui m'a permis de garder une vue d'ensemble sur l'avancement du projet.

Cette implication globale m'a permis de développer des compétences variées et de mieux comprendre la coordination nécessaire dans un projet technique. Malgré les changements de dernière minute, j'ai su m'adapter et continuer à avancer avec le groupe. Ce projet a été une expérience enrichissante et formatrice.

10.2 Thomas

Pendant ce projet, j'ai principalement travaillé sur le code, en particulier l'aspect du site web et du traitement de données récupérées depuis le serveur. J'ai un peu moins travaillé avec les composants physiques car je l'ai plutôt fait vers le début du projet, où nous n'étions pas encore sûrs de la direction que le projet allait prendre.

J'ai dû m'adapter rapidement aux changements de dernière minute que le projet a rencontré mais cela m'a permit d'apprendre à me débrouiller un peu mieux dans ce genre de situations, globalement ce projet était une bonne expérience et avait clairement sa place dans le cours.

10.3 Tristan

Pendant ce projet, j'ai surtout aidé sur la partie électronique : branchements, tests des composants, et mise en place du circuit. J'ai aussi participé à la rédaction des rapport, ce qui m'a permis de bien suivre l'avancée du projet et du groupe.

J'ai moins participé au développement du code (assemblement final et site web), et avec le recul, j'aurais aimé m'impliquer un peu plus dans cette partie. Malgré ça, le projet m'a vraiment appris pas mal de choses, et c'était une bonne expérience de travail.

10.4 Arnaud

Pendant ce projet, j'ai plus travaillé la partie programmation Python. Bien que j'aie participé à la partie électronique, surtout au début du projet lorsqu'il fallait souder les composants sur le PCB. Cela m'a permis d'apprendre à souder correctement ainsi que mieux comprendre la partie électronique.

Le projet ayant changé en dernière minute, il fallait changer le code Python. Mais n'ayant pas constamment accès au matériel, il n'était pas toujours facile de tester le code. Néanmoins, j'ai quand même pu participer aux reconfigurations du code et ce projet m'a donné des opportunités afin de me perfectionner.

10.5 Giorgios

A Codes de test

A.1 Sensor

```
from machine import Pin, time_pulse_us
import time


TRIG_PIN = 17
ECHO_PIN = 18

trig = Pin(TRIG_PIN, Pin.OUT)
echo = Pin(ECHO_PIN, Pin.IN)

def get_distance():
    trig.value(0)
    time.sleep_us(2)

    trig.value(1)
    time.sleep_us(10)
    trig.value(0)

    duration = time_pulse_us(echo, 1, 30000)  # 30 ms timeout

    if duration < 0:
        return None

    distance_cm = (duration * 0.0343) / 2  # Or: duration / 58.0
    return distance_cm

while True:
    distance = get_distance()
    if distance is None:
        print("Out of range")
    else:
        print("Distance: {:.2f} cm".format(distance))
    time.sleep(1)
```

A.2 LEDs

```
from machine import Pin
import time

led_pins = [10, 11, 12, 13]

leds = [Pin(pin, Pin.OUT) for pin in led_pins]

for led in leds:
    led.value(1)

print("LEDs on!")

time.sleep(5)

for led in leds:
    led.value(0)

print("LEDs off!")
```

A.3 LEDs (site active les LEDs)

```
import network
import socket
from machine import Pin
import time

led = Pin(17, Pin.OUT)  # GPIO15

ssid = 'XXXX'   #nom du reseau
password = 'XXXX'  #mdp du reseau

wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(ssid, password)

print("       Connecting to WiFi...")
while not wlan.isconnected():
    time.sleep(1)

ip = wlan.ifconfig()[0]
print("    Connected. Pico W IP Address:", ip)

addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]
s = socket.socket()
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.bind(addr)
s.listen(1)

print("       Listening on http://" + ip)

while True:
    conn, addr = s.accept()
    print("       Client connected from", addr)
    request = conn.recv(1024).decode()
    print("       Request:", request)

    if 'GET /on' in request:
        led.on()
        print("       LED turned ON")
    elif 'GET /off' in request:
        led.off()
        print("       LED turned OFF")

    response = "HTTP/1.1 200 OK\r\nContent-Type: text/plain\r\n\r\nOK"
    conn.send(response)
    conn.close()
```

A.4 Multiplexeur

```
from machine import Pin
import time


bcd_pins = [
    Pin(22, Pin.OUT),  # A (MSB)
    Pin(21, Pin.OUT),  # B
    Pin(20, Pin.OUT),  # C
    Pin(19, Pin.OUT)   # D (LSB)
]

def display_digit(digit):
    if digit < 0 or digit > 9:
        print("Digit must be 0-9")
        return
    binary = [int(x) for x in f"{digit:04b}"]
    for pin, val in zip(bcd_pins, binary):
        pin.value(val)
    print(f"Displayed {digit} -> BCD {binary}")

# cnt 0  9   repeatedly
while True:
    for i in range(10):
        display_digit(i)
        time.sleep(2)
```

A.5 Boutons

```
from machine import Pin
import time

buttons = {
    "A": Pin(8, Pin.IN, Pin.PULL_UP),
    "B": Pin(9, Pin.IN, Pin.PULL_UP),
    "C": Pin(10, Pin.IN, Pin.PULL_UP),
    "D": Pin(11, Pin.IN, Pin.PULL_UP),
}


last_states = {key: pin.value() for key, pin in buttons.items()}

print("       Button test started. Press any button...")

while True:
    for label, pin in buttons.items():
        current = pin.value()
        if last_states[label] == 1 and current == 0:
            print(f"    Button {label} pressed")
        last_states[label] = current
    time.sleep(0.05)
```

A.6 Clavier

```
from machine import Pin
import time

# rows as outputs - active low
rows = [Pin(pin, Pin.OUT) for pin in [7,6,5,4]]

# Columns
cols = [Pin(pin, Pin.IN, Pin.PULL_UP) for pin in [3,2,1,0]]


key_map = [
    ['1', '2', '3', 'A'],
    ['4', '5', '6', 'B'],
    ['7', '8', '9', 'C'],
    ['*', '0', '#', 'D']
]

def scan_keypad():
    for row_index, row in enumerate(rows):
        row.value(0)  # Pull row LOW (active)
        for col_index, col in enumerate(cols):
            if col.value() == 0:  # Pressed key pulls column LOW
                row.value(1)  # Reset row HIGH
                return key_map[row_index][col_index]
        row.value(1)  # Set row HIGH again (inactive)
    return None

while True:
    key = scan_keypad()
    if key:
        print(f"Key Pressed: {key}")
        time.sleep(0.3)  # debounce
```

B Codes finaux

B.1 server.py

```
from flask import Flask, render_template_string, request, jsonify, Response
import requests
import json
import queue

app = Flask(__name__)

# IP of Raspberry Pi Pico (for LED control)
PICO_IP = "http://192.168.129.49"

# Quiz questions
questions = [
    {
        "question": "What is the unit of electrical resistance?",
        "options": {"A": "Ohm", "B": "Watt", "C": "Volt", "D": "Ampere"},
        "answer": "A"
    },
    {
        "question": "What does a capacitor store?",
        "options": {"A": "Heat", "B": "Current", "C": "Energy", "D": "
           Resistance"},
        "answer": "C"
    },
    {
        "question": "Which component restricts the flow of electric current?
           ",
        "options": {"A": "Diode", "B": "Resistor", "C": "Transistor", "D": "
           Capacitor"},
        "answer": "B"
    },
    {
        "question": "What does LED stand for?",
        "options": {"A": "Low Energy Device", "B": "Light Emission Diode", "
           C": "Light Emitting Diode", "D": "Linear Electrical Device"},
        "answer": "C"
    },
    {
        "question": "What kind of current flows in one direction only?",
        "options": {"A": "AC", "B": "DC", "C": "PC", "D": "MC"},
        "answer": "B"
    },
    {
        "question": "Which tool is used to measure voltage?",
        "options": {"A": "Thermometer", "B": "Voltmeter", "C": "Ammeter", "D
           ": "Barometer"},
        "answer": "B"
    },
    {
        "question": "Which device can amplify signals?",
        "options": {"A": "Capacitor", "B": "Transistor", "C": "Resistor", "D
           ": "Fuse"},
        "answer": "B"
    },
    {
        "question": "What does a diode do?",
```

```
        "options": {"A": "Amplifies signal", "B": "Stores charge", "C": "
            Converts AC to DC", "D": "Increases resistance"},
        "answer": "C"
    },
    {
        "question": "Which component protects circuits from overcurrent?",
        "options": {"A": "Resistor", "B": "Inductor", "C": "Transistor", "D"
            : "Fuse"},
        "answer": "D"
    },
    {
        "question": "What does AC stand for?",
        "options": {"A": "Active Current", "B": "Amplified Current", "C": "
            Alternating Current", "D": "Accelerated Charge"},
        "answer": "C"
    },
    {
        "question": "What is the typical voltage of a household battery (AA)
            ?",
        "options": {"A": "1.5V", "B": "5V", "C": "3V", "D": "9V"},
        "answer": "A"
    },
    {
        "question": "Which of these is a passive component?",
        "options": {"A": "Transistor", "B": "Diode", "C": "Resistor", "D": "
            Op-amp"},
        "answer": "C"
    },
    {
        "question": "Which quantity is measured in Amperes?",
        "options": {"A": "Voltage", "B": "Current", "C": "Resistance", "D":
            "Power"},
        "answer": "B"
    },
    {
        "question": "Which material is commonly used as a semiconductor?",
        "options": {"A": "Gold", "B": "Copper", "C": "Silicon", "D": "Iron"
            },
        "answer": "C"
    },
    {
        "question": "What does a breadboard help you do?",
        "options": {"A": "Measure resistance", "B": "Connect circuits
            without soldering", "C": "Store charge", "D": "Amplify current"},
        "answer": "B"
    }
]

# State
current_question_index = 0
selected_answer = None
score = 0

# SSE clients
clients = []

# HTML page


HTML_PAGE = """
```

```
<!DOCTYPE html>
<html>
<head>
    <title>Quiz Bombe</title>
    <style>
        body {
            font-family: monospace;
            background: #181818;
            color: #fff;
            display: flex;
            justify-content: center;
            align-items: center;
            min-height: 100vh;
            margin: 0;
        }
        .quiz-container {
            background: #232323;
            border-radius: 10px;
            padding: 2em;
            width: 500px;
            box-shadow: 0 2px 12px rgba(0,0,0,0.4);
        }
        #question {
            font-size: 1.2em;
            margin-bottom: 1.5em;
            text-align: center;
        }
        #options {
            display: grid;
            grid-template-columns: 1fr 1fr;
            gap: 0.8em;
        }
        .answer {
            background: #333;
            color: white;
            border: none;
            padding: 1em;
            border-radius: 8px;
            cursor: pointer;
            transition: all 0.2s;
        }
        #status {
            font-size: 1.4em;
            color: #fac355;
            margin-bottom: 1em;
            display: block;
            text-align: center;
        }
        @keyframes shake {
            0%, 100% { transform: translateX(0); }
            20%, 60% { transform: translateX(-8px); }
            40%, 80% { transform: translateX(8px); }
        }
    #title {
            position: absolute;
            top: 20px;
            left: 50%;
            transform: translateX(-50%);
            font-size: 2em;
            margin: 0;
```

```
            color: #fac355;
    }
        .hidden { display: none; }
    </style>
    <script>
        async function loadQuestion() {
            const res = await fetch('/question');
            const data = await res.json();
            const qElem = document.getElementById("question");
            const optionsDiv = document.getElementById("options");
            const statusDiv = document.getElementById("status");

            if (data.finished) {
                qElem.textContent = "Quiz Finished! Final Score: " + data.
                    score;
                optionsDiv.innerHTML = "";
                return;
            }

            qElem.textContent = data.question;
            optionsDiv.innerHTML = "";
            for (const [key, value] of Object.entries(data.options)) {
                const p = document.createElement("p");
        p.className = "answer";
                p.textContent = `${key}: ${value}`;
                optionsDiv.appendChild(p);
            }
        }

        const eventSource = new EventSource('/events');
         eventSource.onmessage = function(event) {
    const data = JSON.parse(event.data);

    // 1) If server told us the bomb exploded:
    if (data.game_over) {
      // Show   Game   O v e r  screen
      document.getElementById("question").textContent = "       BOOM! Game
          Over!";
      document.getElementById("options").innerHTML = "";
      document.getElementById("status").textContent = " Y o u ve  reached 3
          strikes.";
      return;
    }

    // 2) Otherwise fall back to your existing logic:
    const status = `You answered ${data.selected}     ${data.result.
       toUpperCase()}<br>Score: ${data.score}`;
    document.getElementById("status").innerHTML = status;
    if (data.next || data.finished) {
      loadQuestion();
    }
  };

        window.onload = loadQuestion;
    </script>
</head>
<body>
    <h1 id="title">Live Quiz</h1>
    <div class="quiz-container">
        <p id="question">Loading...</p>
```

```
        <div id="options"></div>
        <p id="status"></p>
    </div>

</body>
</html>
"""


@app.route("/")
def index():
    return render_template_string(HTML_PAGE)

@app.route("/question")
def get_question():
    global current_question_index
    print(f"avzantline223 {current_question_index}")
    print(f"questionsssz {len(questions)}")
    if current_question_index < len(questions)-1:
        q = questions[current_question_index]
        return jsonify({
            "question": q["question"],
            "options": q["options"],
            "index": current_question_index,
            "score": score
        })
    else :
        print('cacamaca')
        return jsonify({"finished": True, "score": score})

@app.route("/submit_answer", methods=["POST"])
def submit_answer():
    global selected_answer, current_question_index, score
    data = request.get_json()
    if "answer" in data:
        selected_answer = data["answer"]
        correct = questions[current_question_index]["answer"]
        result = "correct" if selected_answer == correct else "incorrect"
        if result == "correct":
            score += 1
        if result == "incorrect":
            score -= 1
        print(f"    Received: {selected_answer}     {result}")

        # Prepare SSE update
        update_data = {
            "selected": selected_answer,
            "result": result,
            "score": score,
            "next": False,
            "finished": False
        }

        # Move to next question
        if current_question_index + 1 < len(questions):
            current_question_index += 1
            update_data["next"] = True
        else:
            update_data["finished"] = True
```

```
278         # Push update to all clients
            for q in clients:
280             q.put(json.dumps(update_data))

282         return jsonify({"status": "ok", **update_data})

284     return jsonify({"status": "error", "message": "Missing answer"}), 400

286 @app.route("/events")
    def events():
288     def event_stream():
            q = queue.Queue()
290         clients.append(q)
            try:
292             while True:
                    data = q.get()
294                 yield f"data: {data}\n\n"
            except GeneratorExit:
296             clients.remove(q)
        return Response(event_stream(), mimetype='text/event-stream')
298
    @app.route("/led/on", methods=["POST"])
300 def led_on():
        try:
302         requests.get(f"{PICO_IP}/on")
            return "LED turned ON! <a href='/'>Go back</a>"
304     except Exception as e:
            return f"Error: {e}"
306 @app.route("/game_over")
    def game_over():
308     # push a game_over event to all clients
        update_data = {"game_over": True}
310     for q in clients:
            q.put(json.dumps(update_data))
312     return jsonify({"status": "ok"})

314 @app.route("/led/off", methods=["POST"])
    def led_off():
316     try:
            requests.get(f"{PICO_IP}/off")
318         return "LED turned OFF! <a href='/'>Go back</a>"
        except Exception as e:
320         return f"Error: {e}"

322 if __name__ == "__main__":
        app.run(host="0.0.0.0", port=5000)
```

B.2 app.py

```
from machine import Pin, time_pulse_us
import network
import time
import urequests
import ujson

#           Wi-Fi setup
ssid = 'Proximus-Home-205507'
password = 'd44sx6x4a7yecppm'
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(ssid, password)
while not wlan.isconnected():
    time.sleep(1)
print("Connected, IP =", wlan.ifconfig()[0])

#           Server URLs
FLASK_SERVER = 'http://192.168.129.15:5000'
# Ensure correct slash before 'sensor'
FIREBASE_URL = (
    'https://anime-time-bomb-default-rtdb.europe-west1.firebasedatabase.app'
    '/sensor/distances.json'
)

#           HC-SR04 sensor setup
TRIG_PIN = 17
ECHO_PIN = 18
trig = Pin(TRIG_PIN, Pin.OUT)
echo = Pin(ECHO_PIN, Pin.IN)

def get_distance():
    trig.value(0)
    time.sleep_us(2)
    trig.value(1)
    time.sleep_us(10)
    trig.value(0)
    duration = time_pulse_us(echo, 1, 30000)
    if duration < 0:
        return None
    return (duration * 0.0343) / 2

#           Firebase push helper
# Use ujson and explicit headers for reliability
def push_distance_to_firebase(distance_cm):
    payload = {'distance_cm': distance_cm, 'ts': int(time.time())}
    try:
        data = ujson.dumps(payload)
        headers = {'Content-Type': 'application/json'}
        resp = urequests.post(FIREBASE_URL, data=data, headers=headers)
        print('Firebase:', resp.status_code, resp.text)
        resp.close()
    except Exception as e:
        print('    Firebase ERROR:', e)

#           Buttons A-D setup
buttons = {
    'A': Pin(8, Pin.IN, Pin.PULL_UP),
    'B': Pin(9, Pin.IN, Pin.PULL_UP),
```

```
    'C': Pin(10, Pin.IN, Pin.PULL_UP),
    'D': Pin(11, Pin.IN, Pin.PULL_UP),
}
last_states = {k: v.value() for k, v in buttons.items()}

#           7-segment display setup
bcd_pins = [Pin(22, Pin.OUT), Pin(21, Pin.OUT), Pin(20, Pin.OUT), Pin(19, Pin.OUT)]
disp1 = Pin(26, Pin.OUT)
disp2 = Pin(27, Pin.OUT)

def display_digit(d):
    bits = [(d >> i) & 1 for i in (3, 2, 1, 0)]
    for pin, b in zip(bcd_pins, bits):
        pin.value(b)

def update_display(score):
    score = min(max(score, 0), 99)
    display_digit(score // 10)
    disp1.value(1); time.sleep(0.003); disp1.value(0)
    display_digit(score % 10)
    disp2.value(1); time.sleep(0.003); disp2.value(0)

#           Heart LEDs (lives)
heart_pins = [13, 14, 15]
hearts = [Pin(p, Pin.OUT) for p in heart_pins]

def update_hearts(lives):
    for i, h in enumerate(hearts):
        h.value(1 if i < lives else 0)

#           Game state
score = 0
lives = 3
update_hearts(lives)

#           Send answer & push sensor on button press
def send_answer(ans):
    global score, lives
    try:
        # 1) Send quiz answer to Flask
        r = urequests.post(f"{FLASK_SERVER}/submit_answer", json={'answer': ans})
        data = r.json()
        r.close()
        print('Flask response:', data)

        # 2) Update score & lives
        score = max(score, data.get('score', score))
        if data.get('result') == 'incorrect':
            lives -= 1
            update_hearts(lives)
            if lives <= 0:
                print('       Game Over')
                try:
                    urequests.get(f"{FLASK_SERVER}/game_over")
                except:
                    pass

        # 3) Read sensor & push to Firebase
```

```
        dist = get_distance()
        if dist is not None:
            print(f'Pushing dist {dist:.2f} cm')
            push_distance_to_firebase(dist)
        else:
            print('Sensor out of range; skipping Firebase push')

    except Exception as e:
        print('Error in send_answer():', e)

#           Main loop
while True:
    for label, pin in buttons.items():
        cur = pin.value()
        if last_states[label] == 1 and cur == 0:
            if lives > 0:
                print('Button pressed:', label)
                send_answer(label)
            else:
                print('Game over     input ignored')
        last_states[label] = cur

    update_display(score)
```