AISC (Artificial Intelligence Smart Cane)

Students Names

Hazem Mohamed
Mohamed Youssef
Begad Tamim
Belal Sameh
Yasser Salem
Abdelrahman Khaled
Mohamed Abdelraheem

College

AAST – College of Artificial Intelligence

Table of Contents

Introduction

Summary

- AISC is a project that aims to aid all blind and visually impaired people worldwide while being affordable, easy-to-use, and easy to build at home using simple electronic components. It really can reshape blind and visually impaired people's lives.

- It helps the user navigate their ways around outdoor obstacles using a lidar and a servo motor, it also helps them locate themselves indoors using NFC chips and a camera that identifies room numbers.

Motives

- We figured that there wasn’t enough awareness for such disability and that visually impaired individuals suffer a lot because of the lack of help they receive despite how fast technology is advancing nowadays.

- To guide visually impaired individuals around without the need of seeking help from strangers.

- Creating the smart cane to be as affordable as possible for the widest range of people while still being produced at the highest quality for its price.

- Fitting the purpose for every visually impaired and blind individual with minimal struggles.

Features

- LIDAR obstacle detection:

  A Tf-mini plus LIDAR is being used for obstacle detection. LIDAR is more accurate than any other ToF sensor. It is used to detect if there is any obstacle that would breakdown the user movement, for example if there is a wall in front of the user the LIDAR is programmed to detect that step and provide an alert to the user, then the wheel at the bottom of the cane redirects the user away from the obstacle which allows him/her to move freely again. The LIDAR is implemented in such a way so that it can perform those steps with all the types of obstacles that would break down the user's movement.

- NFC chips location identifier:

  The smart cane uses NFC reader and tags to detect the user's location.
  This feature is being implemented by using a map for the whole place's location that the cane is going to be used in. Using the map, the NFC stickers will be determined and plotted throughout the whole place. After setting up the NFC stickers locations a data set is collected that relates all those NFC stickers together and depending on the pre scanned NFC the user's orientation and the number of steps that is needed to reach the next point will be calculated.

- Room number recognition camera:

  The raspberry pi HQ camera is placed on top of the board, it is used to live stream the user's movement throughout the whole place. The camera is used in such a way that when the user is in front of a room door the camera will capture the number plate of the room and with the help of computer vision the image is processed and transformed to a text which then will be output to the user.

Hardware Specifications

- Raspberry pi 4b 8gb:

  The Raspberry Pi is a microcontroller that runs on Linux which allows its supported operating system, Pi OS, to be open source and run a suite of open-source software, it also provides a set of GPIO (general purpose input/output) pins, allowing the user to control electronic components for physical computing and Internet of Things (IoT) usage.

Processor	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8)64-bit SoC @ 1.5GHz
Memory	8GB LPDDR4
Connectivity	2.4 GHz and 5.0 GHz IEEE wireless LAN, Bluetooth 5.0, BLE Gigabit Ethernet
GPIO	Standard 40-pin GPIO header
Input power	5V DC via USB-C connector (minimum 3.1 A) 5V DC via GPIO header (minimum 3.1 A) Power over Ethernet (PoE)

- Raspberry Pi HQ Camera:

  The official Raspberry Pi camera.

Sensor	Sony IMX477R stacked, back-illuminated 7.9 mm sensor diagonal
Resolution	12.3 megapixels
HD Video recording	1080p at 30fps, 720p at 60fps, 960p at 45fps
Interface	CSI (Camera Serial Interface)

- RFID-RC522:

  The RC522 RFID module is based on the MFRC522 IC which is the highly integrated RFID card reader/writer IC which works on non-contact 13.56mhz communication. The reader can communicate with a microcontroller over a 4-pin SPI with a maximum data rate of 10 Mbps.

Frequency Range	13.56 MHz ISM Band
Host Interface	SPI / I2C / UART
Operating Supply Voltage	2.5 V to 3.3 V
Max. Operating Current	13-26 mA
Min. Current (Power down)	10 μA
Read Range	5 cm

- Tf-mini Plus:

  The Tf-mini Plus is a single-point short-range, ToF (Time of Flight) LiDAR sensor, to be specific, the Tf-mini Plus emits modulation wave of near infrared ray on a periodic basis, which will be reflected after contacting object.

Operating range	0.1m-12m
Accuracy	±5cm at (0.1-6m ±1% at （6m-12m）
FOV	3.6°
Frame rate	1-1000Hz
Supply voltage	5V±0.5V
Average current	110mA
Peak current	500mA
Communication interface	UART

- L9110 H-Bridge:

  The L9110S is a2-Channel motor driver module is a compact board that has two independent motor driver chips.
  A PWM (Pulse Width Modulation) signal is used to control the speed of a motor and a digital output is used to change its direction.

Chip	2 L9110 motor control chips
Input voltage	2.5-12V DC
Output current	800 mA

- DC Geared Motor:

Power supply voltage	4.5 V
Power consumption	190 mA (max. 250 mA)
Gearbox	48:1
Speed	90 ± 10 rpm
Torque	0.078 Nm

Software Implementation

- Libraries:
  - Pandas: Pandas is a Python library used for working with data sets (i.e., excel sheets). It has functions for analyzing, exploring, and manipulating data. Pandas is used in the RFID code section.
  - Spidev: Spidev is a Python library that helps interface with SPI devices via the Spidev Linux kernel. Spidev is used in the RFID code section.
  - MFRC522: MFRC522 is a Python library used to read/write RFID tags. MFRC522 is used in the RFID code section.
  - Pyttsx3: Pyttsx3 is a text-to-speech conversion library in Python. Pyttsx3 is used in multiple code sections.
  - GPIO: GPIO is a package that provides a Python module to control the GPIO (General Purpose Input Output Pins) on a Raspberry Pi. GPIO is used in multiple code sections.
  - Picamera: Picamera is a Python library designed to provide access to imaging on Raspberry Pi. Picamera is used to capture live video using the Raspberry pi camera. Picamera is used in the Camera code section.
  - Picamera.array: The Picamera library provides a set of classes designed to aid in construction of n-dimensional NumPy arrays from camera output.
  - OpenCV: OpenCV is a Python library that allows you to perform image processing and computer vision tasks. It provides a wide range of features, including object detection, face recognition, and tracking. OpenCV is used in the Camera code section.
  - NumPy: NumPy is a general-purpose array-processing package. It provides a high-performance, multi-dimensional array object and tools for working with these arrays. NumPy is used in the Camera code section.
  - Pytesseract: Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and “read” the text embedded in images. Pytesseract is used in the Camera code section.

- Serial: Serial is a package that gives access to the serial ports for the python code. Serial is used in the LiDAR code section.
- Random: Random is a python library that is used to generate a pseudo-random number. Random is used in the LiDAR code section.

- Camera

```
import cv2
from pytesseract import pytesseract
import pyttsx3
from pytesseract import Output
import time
from picamera.array import PiRGBArray
from picamera import PiCamera
```

Importing necessary libraries.

```
pytesseract.tesseract_cmd = "/usr/bin/tesseract"

engine = pyttsx3.init()

camera = PiCamera()
camera.resolution = (640, 480)
camera.framerate = 32
raw_capture = PiRGBArray(camera, size=(640, 480))

time.sleep(0.1)
```

Initializing the engine that transforms the text to speech. Then we initialize the camera to collect data. The data is then represented in the form of a NumPy array. Then we process the array to extract the words from the array using pytesseract.

```
for frame in camera.capture_continuous(raw_capture, format="bgr", use_video_port=True):
    image = frame.array
    image_data = pytesseract.image_to_data(image,output_type=Output.DICT)
    # cv2.imshow("Frame", image)
    for i,word in enumerate(image_data['text']):
            if word != "":
                print(word)
                engine.say(word)
                engine.runAndWait()

    key = cv2.waitKey(1) & 0xFF
    raw_capture.truncate(0)
```

Cutting the video down to multiple frames.
These frames can be processed to extract the data.
We go frame by frame to extract the words detected and finally, text is processed into speech and said out loud.

- NFC chips location identifier

```
#!/usr/bin/env python
import pandas as pd
import pyttsx3
import RPi.GPIO as GPIO
from mfrc522 import SimpleMFRC522
```

Importing necessary libraries. Telling the raspberry pi that this code is python and that it would be able to be interpreted in the terminal.

```
engine = pyttsx3.init()
reader = SimpleMFRC522()
try:
    id, name = reader.read()
except:
    engine.say("Error while reading")
```

Initializing the text-to-speech engine and creating an object to function as the RFID reader. The reader tries to scan for a tag, unless if an error happens.

```
else:
        nfc_id, name = reader.read()
        our_id = name[len(name) - 2] + name[len(name) - 1]
        if our_id.isnumeric() == True:
            our_id = int(our_id)
```

```
if 1 <= our_id <= 25:
    dataframe = pd.read_excel("/home/harry/Cane/RFID/Building_NFC_Dataset_Code_Format.xlsx")
    dfcolumn = dataframe.iloc[our_id - 1]
    column = dfcolumn.values.tolist()  # type: ignore
    column_free=[i for i in column if i!="&"]
```

When the reader scans a tag, the tag's name and ID are obtained and then derive the desired ID from the obtained ID, and then confirm that the ID is an integer. The tag gets verified if it is one of our written tags.
Then we translate the excel datasheet to a data frame object, get the specified column using our ID, and finally fill the values of the column into a list. The list is then cleared from any empty indices which are identified with '&'.

```
if pre == NULL:
    at = column[1]
    at = at[:-3]
    text = "You are at " + at
    engine.say(text)
    engine.runAndWait()
    if column[2] != "&":
        text = "To Your East is " + column[2] + "By " + str(column[3]) + " Steps"
        engine.say(text)
        engine.runAndWait()
    if column[4] != "&":
        text = "To Your West is " + column[4] + "By " + str(column[5]) + " Steps"
        engine.say(text)
        engine.runAndWait()
    if column[6] != "&":
        text = "To Your North is " + column[6] + "By " + str(column[7]) + " Steps"
        engine.say(text)
        engine.runAndWait()
    if column[8] != "&":
        text = "To Your South is " + column[8] + "By " + str(column[9]) + " Steps"
        engine.say(text)
        engine.runAndWait()
```

"Pre" is a variable that stores the previous tag that was read. It helps the cane to determine its orientation with respect to the scanned tag. If the pre is NULL, then the orientation is not specified. Therefore, it would tell the user his NWES orientation according to the read tag.

```
                pre = our_id
finally:
        GPIO.cleanup()
```

After every scan, the read tag is then stored as our Pre tag. Finally, the buffer is cleaned up to accept new data without any data clattering.

- LIDAR obstacle detection

```
#!/usr/bin/env python
import random
import pyttsx3
import RPi.GPIO as GPIO
import serial
```

Importing necessary libraries. Telling the raspberry pi that this code is python and that it would be able to be interpreted in the terminal.

```
GPIO.setmode(GPIO.BCM)
pinLeft = 31
pinRight = 32
GPIO.setup(pinLeft, GPIO.OUT)
GPIO.setup(pinRight, GPIO.OUT)
engine = pyttsx3.init()
```

Setting the designated GPIO pins that will be used by the motors and controlled by the LiDAR. Then initializing our text-to-speech engine that will help guide the user.

```
ser = serial.Serial("/dev/ttyAMA0", 115200)
left = GPIO.PWM(pinLeft, 50)
right = GPIO.PWM(pinRight, 50)
distance_threshold = 70
strength_threshold = 100
    rand_direction = random.randint(0, 1)
    min_speed = 0
    max_speed = 30
```

Enabling the serial port and setting the frequencies of the H-bridges that will control the motors.
Distance will be set to a maximum threshold of 70 centimeters and strength will be set to a minimum threshold of 100.
A random integer will then be generated to decide if the cane will move left or right depending on the surroundings.
The motor's speed are capped at 30% duty cycle.

```
def read_data():
    while True:
        counter = ser.in_waiting()
        if counter > 8:
            bytes_serial = ser.read(9)
            ser.reset_input_buffer()
if bytes_serial[0] == 0x59 and bytes_serial[1] == 0x59:
    distance = bytes_serial[2] + bytes_serial[3] * 256
    strength = bytes_serial[4] + bytes_serial[5] * 256
    # temperature = bytes_serial[6] + bytes_serial[7] * 256
    # temperature = (temperature/8) - 256
```

A function "read_data" will then be defined to read if the LiDAR is on. When reading from the LiDAR, 9 values are obtained. When the needed values are obtained, the values are then added to a list to be manipulatable and finally we clean up the buffer.
The two values of the list are the starting addresses. Distance, strength, and temperature are calculated using the bits (2,3), (4,5) and (6,7) respectively. The last value is the ending address.

```
if strength >= strength_threshold and distance <= distance_threshold:
    text = "There is an obstacle  " + str(distance) + " centimeters to your front"
    engine.say(text)
if rand_direction == 1:
    left.start(min_speed)
    for i in range(min_speed,max_speed):
        left.ChangeDutyCycle(i)
        if strength >= strength_threshold and distance > distance_threshold:
            left.stop()
            break
    left.stop()
elif rand_direction == 0:
    right.start(min_speed)
    for i in range(min_speed,max_speed):
        right.ChangeDutyCycle(i)
        if strength >= strength_threshold and distance > distance_threshold:
            right.stop()
            break
    right.stop()
```

A condition then compares the strength and distance of the object with respect to the LiDAR and warns the user if an object was detected using text-to-speech. Using the random generated number, the cane goes to the decided direction with gradual speed until the object is no longer detected.

```
    GPIO.cleanup()
ser.reset_input_buffer()
if __name__ == "__main__":
    try:
        if ser.isOpen() == False:
            ser.open()
        GPIO.cleanup()
        read_data()
    #except KeyboardInterrupt: # ctrl + c in terminal.
        #if ser != None:
        #    ser.close()
        #GPIO.cleanup()
        #print("Program interrupted by the user")
    finally:
        GPIO.cleanup()
```

Buffers are then cleaned up to avoid any data interferences and then we end the function. Going to the main function, the LiDAR is then turned on and starts reading data using the function “read_data” then the buffers clear. Finally, the GPIO buffer is cleaned up to accept new data without any data clattering.

Project Timeline

	Description of Work	Start and End Dates
Phase One	Brainstorming	20/10/2022 till 23/10/2022
Phase Two	Developing the Code and Setting up the Hardware	24/10/2022 till 7/11/2022
Phase Three	Code debugging and Project Testing	8/11/2022 till 14/11/2022

Future Plans

- To integrate the spectacles 3 by using the glasses' cameras for object detection.
- Developing a mobile app that:
  - Notifies the users family and friends of the user's whereabouts.
  - Uses the user's phone for audio I/O.
  - Using google street/blind square view for more accurate outdoor usage.
- Adding more languages to the audio I/O.
- Implementing a heart rate sensor which monitors the user's health condition.
- Updating the software and the hardware continuously.

References

- https://www.w3schools.com/python/pandas/pandas_intro.asp
- https://pypi.org/project/spidev/
- https://pypi.org/project/mfrc522/
- https://pypi.org/project/RPi.GPIO/
- https://pypi.org/project/pyttsx3/
- https://www.americangeosciences.org/critical-issues/faq/what-lidar-and-what-it-used#:~:text=LIDAR%20can%20also%20be%20used,%2C%20mining%2C%20and%20much%20more
- https://pimylifeup.com/raspberry-pi-rfid-rc522/
- https://automaticaddison.com/how-to-set-up-real-time-video-using-opencv-on-raspberry-pi-4/
- https://www.mygreatlearning.com/blog/opencv-tutorial-in-python/
- https://www.geeksforgeeks.org/numpy-in-python-set-1-introduction/#:~:text=What%20is%20NumPy%3F,for%20scientific%20computing%20with%20Python
- https://picamera.readthedocs.io/en/release-1.13/api_array.html
- https://pypi.org/project/pytesseract/
- https://www.geeksforgeeks.org/python-time-module/
- https://pythonhosted.org/pyserial/