AIoT and Cloud Enabled Flexible and Reliable Monitoring Platform

Carly Ray, Kyle Mooney, Jinhui Wang*
Electrical and Computer Engineering
University of South Alabama Mobile, AL, USA
jwang@southalabama.edu

Grayson Dennis
Chemical Engineering
University of South Alabama
Mobile, AL, USA

Shenhua Wu
Civil, Coastal, and Environmental Engineering
University of South Alabama
Mobile, AL, USA

***Abstract*—In this paper, an AIoT and Cloud enabled monitoring platform is proposed. The platform includes Cloud, sensor nodes, coordinators with different communication protocols, networks, portable Artificial Intelligence (AI) accelerators, and acrylonitrile styrene acrylate (ASA) case using 3D printing. Sensor nodes and communication protocols can be selected and adapt to any application. In our developed monitoring platform prototype, WiFi, Cellular (LTE, GSM, CDMA, etc.) and Zigbee protocols are utilized for environmental and object monitoring, as well as data analysis and display. The experimental results demonstrate (1) detailed environmental information display in 9 locations and (2) much faster and more accurate object monitoring with the portable AI accelerator.**

***Index Terms*—AIoT, Could, accuracy, environmental monitoring system, code, sensor node, coordinator, 3D printing, WiFi, cellular, Zigbee**

I. Introduction

Today, with connectivity becoming increasingly necessary in our daily work and social lives, more and more devices are connected to the Internet and each other more than ever, which makes IoT (Internet of Things) and Cloud systems significant players in the evolving world of technology [1], [2].

IoT refers to devices connected to the Internet, and then current and historical data are stored and processed in real time in the IoT. Artificial intelligence (AI) algorithms are allowed to run in IoT systems to simulate a level of human intelligence for more effective data processing and higher computing capabilities [3], [4]. Such a combination of AI and IoT, named AIoT, enables the end nodes on a connected network to make intelligent decisions using the collected data. Therefore, AIoT can gather large and widespread data, while being benefited from AI [5].

Cloud is a centralized system to help transfer and deliver data and files to data centers over the Internet. The Cloud runs exclusively on the Internet rather than on a local machine, allowing large amounts of data to be stored in a central location that multiple/many devices can access at the same time [6]. The AIoT and Cloud computing are two different concepts, but they often work together. Data from the end nodes of an AIoT system can be sent to a Cloud, which processes the data through either AI or other algorithms.

This work was supported in part by the National Oceanic and Atmospheric Administration under Grant NA23OAR4170169 and the National Science Foundation under Grants 2218046 and 2247343.

In this paper, a flexible monitoring platform that utilizes both AIoT and Cloud is proposed. The system utilizes a Wireless Sensor Network (WSN) in IoT to collect monitoring data incrementally. The data then are sent to the Cloud for processing, storing, organizing, and displaying. This platform is flexible for different purposes, since sensors can be changed based on specific applications, such as environmental monitoring, healthcare monitoring, and agriculture monitoring. In urban areas, this monitoring platform uses easily accessible WiFi or Cellular networks for communication. When it is in rural areas, it transfers the data through node-by-node communication, such as the Zigbee protocol, so it is a 24/7 monitoring and all-weather platform.

A. Related Work

Many researchers work on different IoT and Colud enabled monitoring systems [7]–[18]. **General solution**. In [19], [20], a solution is proposed to collect raw data from the distributed sensor nodes in IoT to speed up processing. In [21], [22], the Cloud and IoT are improved for custom-designed data storage and processing to enable privacy preservation. In [23], an updated architecture is proposed to embed WSN into the Cloud for fast data collection and storage. In [24]–[32], an emerging device, the memristor, is manufactured and used for low-power and high-performance Edge AI applications and for privacy preserving in IoT using the inherent nonlinearity and variations of memristors. **Specific Applications**. In terms of environmental monitoring, numerous exploratory investigations have been conducted. In [33], a holistic IoT-based management platform is proposed for environmental monitoring. An environmental monitoring system is proposed for Smart Cities in [34]. In [17], an outdoor air quality monitoring system is created based on the ZigBee WSN. Regarding Cloud- and IoT-based healthcare monitoring, [35] explores the fundamentals of IoT for remote diagnosis and healthcare applications; and [36] focuses on security of the IoT devices that store the data related with healthcare issues. As for agriculture monitoring, a significant amount of work has been completed. In [37], a smart greenhouse design is implemented to eliminate manual intervention and measure different climate parameters by sensors in AIoT. In [38], [39], Cloud based software and new algorithm have been proposed

979-8-3503-6275-6/24/$31.00 ©2024 IEEE

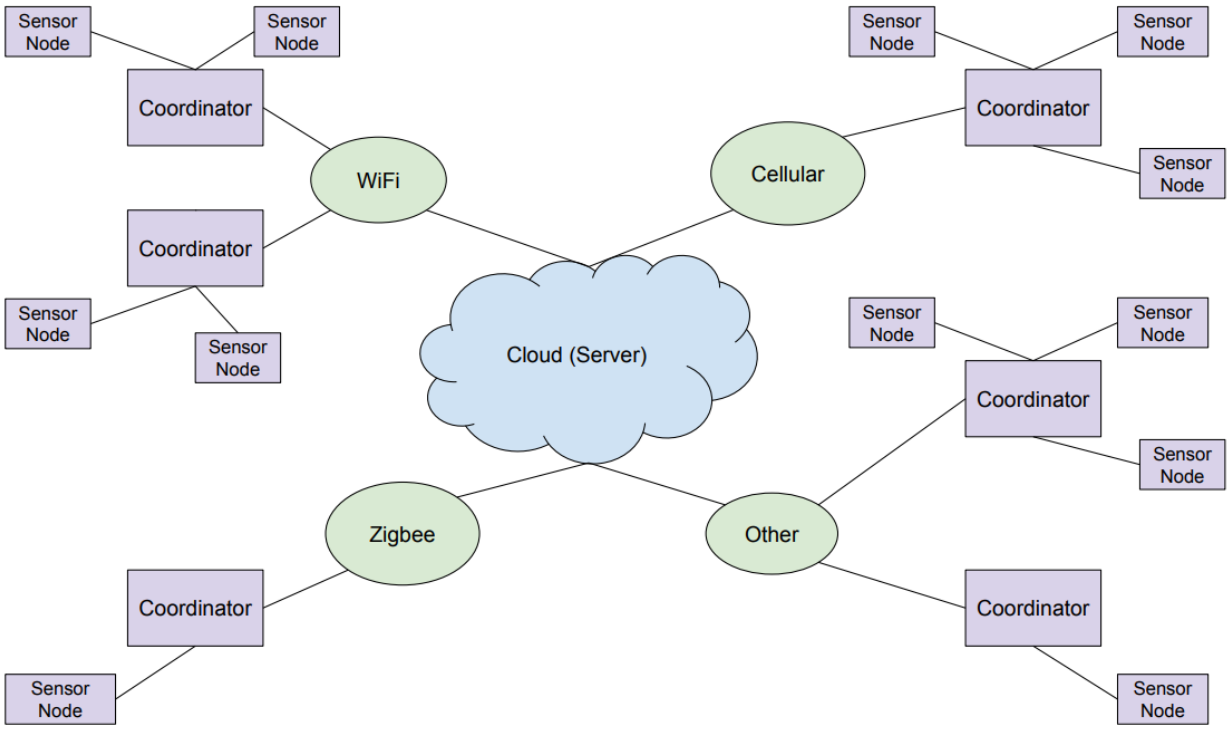


Fig. 1. AIoT and Cloud Enabled Flexible Monitoring Platform.

to much more accurately process and retrieve information and agricultural tasks.

However, most of the above systems are limited to a specific application, not universal. Therefore, a flexible and reliable platform is needed to be compatible with any application and demand. As a result, a novel platform is proposed in this paper.

B. Contributions

Specifically, this paper makes the following contributions: (1) The detailed introduction for an AIoT and Cloud enabled flexible and reliable monitoring platform is given. (2) A monitoring platform prototype is developed for the demonstration. (3) Two applications, environmental monitoring and object monitoring, are described to show the flexible and reliable of the platform.

II. Proposed Platform

A. Network and System

The proposed AIoT and Cloud enabled flexible monitoring platform is made up of several components as shown in Figs. 1 and 2: Cloud, sensor nodes, coordinators with different communication protocols, portable Artificial Intelligence (AI) accelerators, and acrylonitrile styrene acrylate (ASA) case using 3D printing. The central part of the system is the Cloud. As shown in Fig. 1, no matter which type of communication protocol is utilized in each branch, all the data is collected into the Cloud, where the data is processed and organized. A website and map are created to display all the data stored on the Cloud in real-time. When the data are received by the Cloud, the website and map can show the ID, time, and location of the sensor node, data information, and even analysis results with AI algorithm.

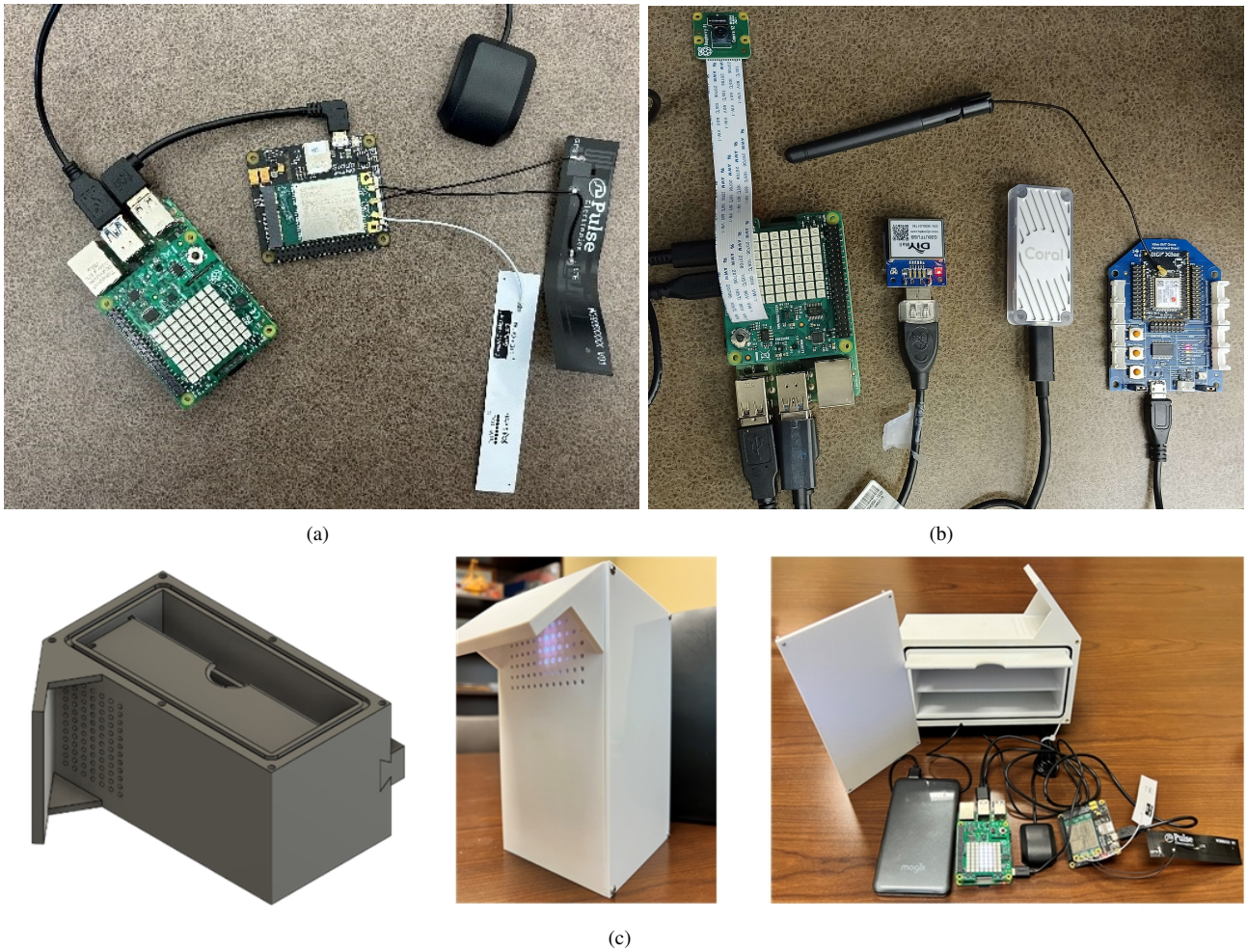
The edges of the platform are the coordinators and sensor nodes. Sensors can be different types based on the specific application. There can be numerous sensor nodes connecting to a coordinator with different protocols. The number of coordinators and sensor nodes is adjustable and determined by the size of the monitoring area, data density, and maintenance cost. In our developed monitoring platform prototype, the Google Cloud is employed. The coordinators and node controllers are chosen as a Raspberry Pi 4B. The sensor is a SenseHAT, which gathers the temperature, humidity, and barometric pressure data to display on the map, as shown in Figs. 1 and 2.

B. Communication Protocols

As also shown in Fig. 1, there are 3 communication protocols for data transfer in the monitoring platform, including WiFi, Cellular (LTE, GSM, CDMA, etc.), and Zigbee. It indicates that different types of communication protocols can be used according to the project need. For indoor and urban areas, WiFi or Cellular are good choices for easy access to send/receive the data to/from the Cloud. In remote areas, node-to-node communication, such as Zigbee, can be used instead of WiFi or Cellular. Actually, communication protocols are not limited to above 3 types, it can be extended to any others, depending on locations, environment, cost, and etc.

C. Accelerator

In our developed monitoring platform prototype, Edge devices can be used to improve the processing capabilities of the system by intelligently collecting data rather than linearly



(a) (b)

(c)

Fig. 2. (a) Monitoring Platform with Cellular Modem. (b) Monitoring Platform with Zigbee block and AI accelerator. (c) (From left to right) Autodesk Fusion 360 Orthographic View of Case Design; 3D Printed Model with Electronics within Case; and Electronics (Raspberry Pi, GPS, LTE modem, and battery) Contained within Weatherproof Case.

collecting. An example of the usage of an Edge device would be in image processing. For example, if a model is trained and used at the coordinator or sensor node of the system for recognition or detection, the data can be organized and pre-filtered and there will be less traffic in the data communication, since only the correct and selected image will be sent to the Cloud. Therefore, in a lightweight and mobile system, the processing capabilities of the coordinator and sensor node are an important characteristic to be taken into account. In our developed monitoring platform prototype, a Raspberry Pi 4B is used as coordinator, but it struggles to run large AI models. Therefore, an accelerator is needed. As shown in Figs. 2 (b), a sensor node with a Google Edge TPU coprocessor (Google Coral Accelerator: 4 TOPS and 2 TOPS per watt) is used to accelerate AI computation on the Edge device.

D. ASA Case Using 3D Printing

The software, Autodesk Fusion 360, is used to design the apparatus of the ASA case. The main design criteria for the case are: (1) weather resistance, (2) electronics compartment, and (3) aesthetics. Criteria (1) is the primary issue. Due to the sensitivity of the equipment, weather in the region could cause a disruption of data acquisition, leading to inadequate data records. The addition of a main compartment allows for contact between the Raspberry Pi and the environment to collect air quality data. A gasket-filled lid creates a weather-resistant seal along with heat inserts to secure the lid. Finally, the addition of an angled roof where air circulation occurs ensures that air can adequately contact the Raspberry Pi while also protecting from nearly all elements (i.e., rain, wind, hail, etc.). These elements satisfy the requirement of the criteria (1). The Criteria (2) is much easier to accommodate. A compartment is made to allow the Raspberry Pi with Sensor hat to contact the environment while allowing easy access to all the components in the case that batteries need to be changed or the Raspberry Pi needs to be debugged. The Criteria (3) is accomplished once both Criteria (1) and (2) are completed. As Fig. 2 (c) demonstrates, the case looks like a bird house, so when mounted in an urban, suburban, or rural area, it will look inconspicuous to the layman. The design should help deter tampering while looking more aesthetically pleasing. Finally, the case material is acrylonitrile styrene acrylate

(ASA) sourced from the overture. The material is chosen due to its ultraviolet (UV) resistance and impact resistance, as well as white color. All of these should help ensure the longevity of the case as well as keep the Raspberry Pi from overheating.

III. Experimental Results and Analysis

A. Important Codes

In our developed monitoring platform prototype, a ZigBee mesh network is used to send/receive data between sensor nodes and coordinators, when WiFi and cellular networks are not available. Firstly, sensor nodes collect data and send their data to the nearby coordinator. The coordinator in turn then sends the data to the Cloud in *csv* file format through the ZigBee network. The codes for setting up the ZigBee modules on all nodes, the ZigBee code inside the main loop of coordinators, and the ZigBee code inside the main loop of sensor nodes, are shown in Figs. 3, 4, and 5.

```
#setup ZigBee device, open the port, and get devices on the same network
device = XBeeDevice('/dev/ttyUSB0', 9600)
device.open()
xbee_network = device.get_network()
```

Fig. 3. Setting Up ZigBee Modules on All Sensor Nodes.

```
# Receive data from another device on ZigBee network
   net_message = device.read_data()
   if (net_message != None): #if message received; decode it
       net_message = net_message.data.decode() + f",'Time':
{datetime.now()}"
       print(net_message)
```

Fig. 4. ZigBee Code Inside Main Loop of Coordinators.

```
# Formatting data for csv file on Coordinator node
message =
f"'ID':{node_name},'Temp':{temp},'Hum':{humidity},'Pressure':{pressure},'L
at':{lat},'Long':{long}"
# Send data to all devices listening on ZigBee network
device.send_data_broadcast(message)
```

Fig. 5. ZigBee Code Inside of Main Loop of Endpoint Nodes.

Since the Raspberry Pi is used on our platform as coordinators or controllers, it has an attached Cellular modem to give the ability to use cellular communication, as shown in Fig. 2 (a). The device will be configured to see whether WiFi is available or not. If the Wifi is not ready, the system would switch to Cellular communication to send/receive the data. The coordinator and sensor nodes can be exchanged according to the need of the data and the role in the project, allowing flexibility and versatility of the platform.

As mentioned previously, the devices and networks on the platform are connected to the Cloud. The code is shown in Fig. 6. Credentials are necessary to provide security to the application, so the key is passed into the program from the device. A bucket is a container to hold the objects needed for the project, and a blob is an object that is inside the bucket on the Cloud. In this section of the code, we are accessing the necessary container and listing the objects inside so that the needed blob can be found and uploaded.

Position data is important in our developed monitoring platform prototype, as it is necessary to display the data on the map for analysis. Position data is collected using the GPS module that is externally connected to the Raspberry Pi and collected in the program using the code in Fig. 7. These position data are then combined with sensor data collected using the SenseHAT, and added to a *csv* file. This *csv* file is finally uploaded to the Cloud. This process repeats every hour (if the data is collected every hour), as in the code in Fig. 8.

```
#variables and declarations used for cloud
#set directory where the cloud credentials/key are
os.environ["GOOGLE_APPLICATION_CREDENTIALS"]= "key.json"
store = storage.Client()
#lists all available buckets
buckets = list(store.list_buckets())
#use the bucket you named in Cloud storage
bucket = store.get_bucket("iotenabledsmartenvmonitoring.appspot.com")
blobs = store.list_blobs("iotenabledsmartenvmonitoring.appspot.com")
```

Fig. 6. Setting for Google Cloud.

```
def get_Coord(): #function to retrieve GPS coordinates
    #gps= serial.Serial("/dev/ttyACM0")
    session = gps.gps(mode=gps.WATCH_ENABLE)
    while 0 == session.read():
        if not (gps.MODE_SET & session.valid):
            continue

        print('Mode: %s(%d) Time: ' %
              (("Invalid", "NO_FIX", "2D", "3D")[session.fix.mode],
               session.fix.mode), end="")
        # print time, if we have it
        if gps.TIME_SET & session.valid:
            print(session.fix.time, end="")
        else:
            print('n/a', end="")

        if ((gps.isfinite(session.fix.latitude) and
             gps.isfinite(session.fix.longitude))):
            lat = session.fix.latitude
            long = session.fix.longitude
            print(lat)
            print(long)
            #print(" Lat %.6f Lon %.6f" %
             #     (session.fix.latitude, session.fix.longitude))
        else:
            print(" Lat n/a Lon n/a")
        coord = f"{lat},{long}"
```

Fig. 7. Collecting Position Data.

```
#loop for getting environmental data from the pi, and uploading it to the cloud
while True:
    temp = sense.get_temperature()
    temp = ((temp/5)*9)+32
    pressure = sense.get_pressure()
    humidity = sense.get_humidity()

    node_name = f'{name}_{count}'

    #Puts environmental data into a string
    coords = get_Coord()
    lat = coords.split(",")[0]
    long = coords.split(",")[1]

    csvReader(dataLines,temp,pressure,humidity,node_name, lat, long)
    #opening created csv file to append data to a new row in it

    #uploading the csv file to Google Cloud
    blob = bucket.blob(f"data/{csv_name}")
    recv = blob.upload_from_filename(f"{csv_name}")
    count = count + 1

    #pause of "x" amount of seconds
    time.sleep(3600)
    #time.sleep(60)
```

Fig. 8. Uploading Data to Cloud.

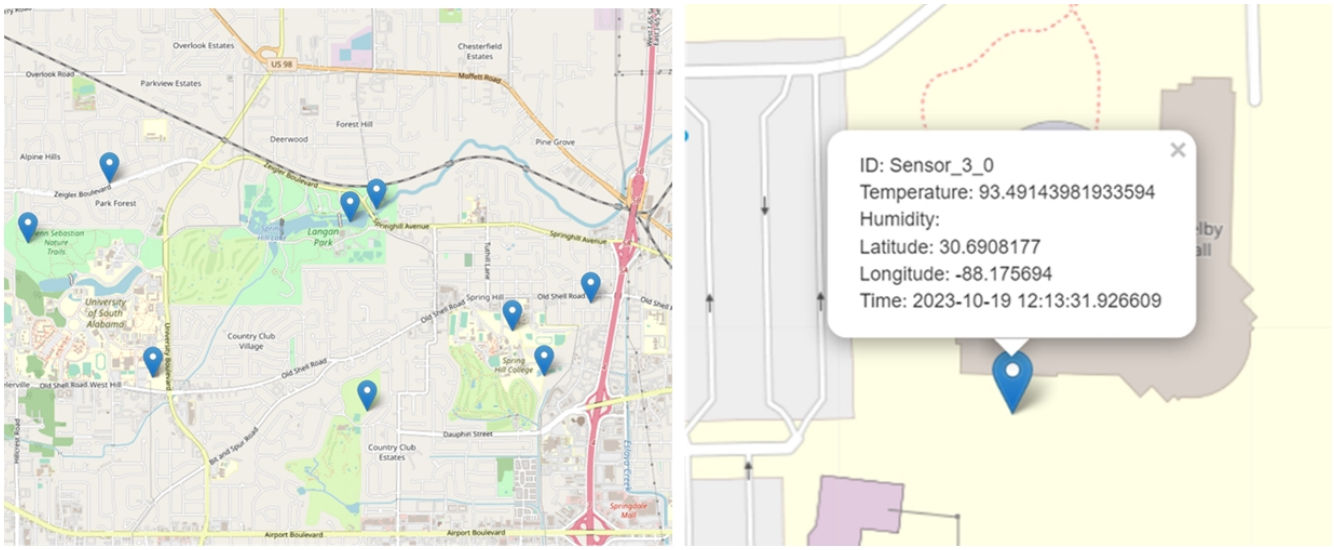


Fig. 9. Locations and Information of Sensor Nodes.

TABLE I
DATA FROM SENSOR NODES

ID	Date Time	Temperature (F)	Humidity (rH)	Barometric Pressure (Millibars)	Latitude	Longitude
Sensor_2_0	12/5/2023 20:24	70.48384864	84.98821176	1006.657791	30.69452008	-88.12016946
Sensor_2_1	12/5/2023 21:24	70.13550237	84.42703784	1008.140927	30.69784988	-88.16305326
Sensor_2_2	12/5/2023 22:24	71.98952597	83.27079063	1013.168887	30.70569881	-88.10140890
Sensor_2_3	12/5/2023 23:24	70.62995911	84.07781890	1013.004590	30.70350172	-88.14151975
Sensor_2_4	12/5/2023 0:24	70.88124208	83.79608197	1007.199209	30.69927027	-88.13736474
Sensor_2_5	12/5/2023 1:24	71.32157650	83.71916639	1000.208420	30.68967268	-88.18848276
Sensor_2_6	12/5/2023 2:24	71.77213321	84.77007491	1003.991880	30.69652664	-88.16415512
Sensor_2_7	12/5/2023 3:24	70.40468789	84.23183246	1012.949528	30.70426642	-88.12201916
Sensor_2_8	12/5/2023 4:24	71.26431357	83.82901174	1000.208169	30.70581310	-88.13836647

B. Experimental Results

As mentioned in previous sections, our developed monitoring platform prototype uses a Raspberry Pi as a core device. A SenseHAT is used as the sensor in each node, and it collects temperature, humidity, and Barometric pressure data in real time. A GPS module is attached to the core device, and position data would be gathered along with the sensor data. Cloud is used in our platform. All of this data as well as a node ID and time information are collected into a line on a *csv* file, and then the *csv* file would be uploaded to the Cloud storage. The Cloud storage contains the code for the website, which cycles through the available *csv* files and displays them using Flask. Flask codes are shown in Fig. 10. And the monitoring results are shown in Fig. 9. Because the type of data chosen is temperature, humidity, and barometric pressure, a screenshot of the monitoring website and map indicates 9 locations of the sensor nodes with data inforamtion. The data is captured every hour and listed in the Table I with ID, time, and location of the sensor node.

Our developed monitoring platform prototype with an image sensor and Coral accelerator is also tested for an object monitoring application. A model is trained using the COCO dataset, which is converted into a TFLite model [40] and lightweight enough to function on smaller processing devices such as the Raspberry Pi with AI accelerator. A code shown in Fig. 11 is then created to process a live stream video feed to be able to highlight detected objects. The experimental results are shown in Fig. 12. Not only does the TFLite provide a lighter-weight model for processing, but an accelerator also boosts the

```
def home():
    previous_entry = ""
    ID_list = []
    datalist = []
    for file in file_list:
        try:
            print(file)
            fileObject = urllib.request.urlopen(url + file)
            data = fileObject.read().decode('utf-8')
            fileObject.close()
            csv_reader = csv.DictReader(data.split('\n'), delimiter = ',')
            #print(csv_reader)
            #for row in reversed(list(csv_reader)):
            for row in list(csv_reader):
            #print(previous_entry)
                print(row['ID'])
                if row['ID'] == "ID":
                    continue
                else:
                    ID_list.append(row['ID'])
                    datalist.append(row)
                    #print(ID_list)
                    #print(datalist)
        except IOError:
            print("File already opened")
            continue

    return render_template('home.html', nodes = datalist)
@app.route('/about')
def about():
    return render_template('about.html')
```

Fig. 10. Flask Code.

```
# Perform the actual detection by running the model with the image as input
interpreter.set_tensor(input_details[0]['index'],input_data)
interpreter.invoke()

# Retrieve detection results
boxes = interpreter.get_tensor(output_details[0]['index'])[0] # Bounding box coordinates of detected objects
classes = interpreter.get_tensor(output_details[1]['index'])[0] # Class index of detected objects
scores = interpreter.get_tensor(output_details[2]['index'])[0] # Confidence of detected objects
#num = interpreter.get_tensor(output_details[3]['index'])[0]  # Total number of detected objects (inaccurate and not needed)

# Loop over all detections and draw detection box if confidence is above minimum threshold
for i in range(len(scores)):
    if ((scores[i] > min_conf_threshold) and (scores[i] <= 1.0)):

        # Get bounding box coordinates and draw box
        # Interpreter can return coordinates that are outside of image dimensions, need to force them to be within image using max() and min()
        ymin = int(max(1,(boxes[i][0] * imH)))
        xmin = int(max(1,(boxes[i][1] * imW)))
        ymax = int(min(imH,(boxes[i][2] * imH)))
        xmax = int(min(imW,(boxes[i][3] * imW)))

        cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0), 2)

        # Draw label
        object_name = labels[int(classes[i])] # Look up object name from "labels" array using class index
        label = '%s: %d%%' % (object_name, int(scores[i]*100)) # Example: 'person: 72%'
        labelSize, baseLine = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.7, 2) # Get font size
        label_ymin = max(ymin, labelSize[1] + 10) # Make sure not to draw label too close to top of window
        cv2.rectangle(frame, (xmin, label_ymin-labelSize[1]-10), (xmin+labelSize[0], label_ymin+baseLine-10), (255, 255, 255), cv2.FILLED) # Draw white box to put label text in
        cv2.putText(frame, label, (xmin, label_ymin-7), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2) # Draw label text

# Draw framerate in corner of frame
cv2.putText(frame,'FPS: {0:.2f}'.format(frame_rate_calc),(30,50),cv2.FONT_HERSHEY_SIMPLEX,1,(255,255,0),2,cv2.LINE_AA)

# All the results have been drawn on the frame, so it's time to display it.
cv2.imshow('Object detector', frame)

# Calculate framerate
t2 = cv2.getTickCount()
time1 = (t2-t1)/freq
frame_rate_calc= 1/time1
```

Fig. 11. Code for Object Monitoring.

(a) (b)

Fig. 12. Before (a) and After (b) Adding Accelerator to Platform for Object Monitoring.

processing speed and capability enough to process data in real time. As listed in the Table II, the Frames Per Second (FPS) of the live stream is improved by approximately 4 times after AI accelerator is added to the Raspberry Pi and accuracy is also much improved. Especially for Keyboard monitoring, it has 20% accuracy increase.

TABLE II
TESTED RESULTS BEFORE AND AFTER AI ACCELERATOR IS ADDED

FSP	Accuracy (%)	TV	Bottle	Keyboard
Before=4.27	Before	71%	66%	78%
After=16.03	After	78%	66%	98%

CONCLUSION

An AIoT and Cloud enabled monitoring platform is proposed in this paper. Environmental and object monitoring is demonstrated to verify the flexibility and reliability of the platform. Especially, with the AI accelerator, the (FPS) of the live stream is improved by approximately 4 times and the inference accuracy increases by up to 20%.

ACKNOWLEDGMENT

This work was supported in part by the National Oceanic and Atmospheric Administration under Grant NA23OAR4170169 and the National Science Foundation under Grants 2218046 and 2247343. The authors acknowledge financial support from the Alabama Graduate Research Scholars Program (GRSP), funded through the Alabama Commission for Higher Education and administered by the Alabama EPSCoR.

REFERENCES

[1] M. Oli-Uz-Zaman, S. A. Khan, W. Oswald, Z. Liao, and J. Wang, "Stuck-at-fault immunity enhancement of memristor-based edge ai systems," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 12, no. 4, pp. 922–933, 2022.

[2] L. Hou, F. Fan, J. Fu, and J. Wang, "Time-varying algorithm for swarm robotics," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 217–222, 2017.

[3] J. Oh, J. Lee, Y. Park, and Y. Park, "A secure data processing system in edge computing-powered aiot," in *2022 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, 2022, pp. 1–4.

[4] J. Edstrom, Y. Gong, D. Chen, J. Wang, and N. Gong, "Data-driven intelligent efficient synaptic storage for deep learning," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 12, pp. 1412–1416, 2017.

[5] C. González García, E. R. Núñez Valdéz, V. García Díaz, B. C. Pelayo García-Bustelo, J. M. Cueva Lovelle *et al.*, "A review of artificial intelligence in the internet of things," *International Journal Of Interactive Multimedia And Artificial Intelligence, 5*, 2019.

[6] J. Shen, T. Zhou, D. He, Y. Zhang, X. Sun, and Y. Xiang, "Block design-based key agreement for group data sharing in cloud computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 6, pp. 996–1010, 2017.

[7] R. Ge, H. Pan, Z. Lin, L. Hou, N. Gong, and J. Wang, "Rf-powered battery-less wireless sensor network," in *2016 5th International Symposium on Next-Generation Electronics (ISNE)*, 2016, pp. 1–3.

[8] Y. Gong, Z. Lin, J. Wang, and N. Gong, "Bringing machine intelligence to welding visual inspection: Development of low-cost portable embedded device for welding quality control," *Electronic Imaging*, vol. 30, pp. 1–4, 2018.

[9] S. Geng, H. Yang, C. Liu, L. Hou, and J. Wang, "A design of the node system of wireless sensor net for ancient building fire prevention," in *2012 Asia-Pacific Symposium on Electromagnetic Compatibility*, 2012, pp. 241–244.

[10] H. Pan, R. Ge, J. Wang, N. Gong, and Z. Lin, "Integrated wireless sensor networks with uas for damage detection and monitoring of bridges and other large-scale critical civil infrastructures," *NDE/NDT for Highway and Bridges: Structural Materials Technology, Portland, OR, USA*, 2016.

[11] W. Zhongchao, H. Ligang, T. Baojun, W. Wensi, and W. Jinhui, "Design and verification of a novel iot node protocol," in *2017 13th IEEE International Conference on Electronic Measurement & Instruments (ICEMI)*, 2017, pp. 201–205.

[12] R. Ge, H. Pan, Z. Lin, N. Gong, J. Wang, and X. Chen, "Rf-powered battery-less wireless sensor network in structural monitoring," in *2016 IEEE international conference on electro information technology (EIT)*, 2016, pp. 0547–0552.

[13] S.-q. Geng, L.-g. Hou, J.-h. Wang, L. Zuo, W. Zhang, and W.-c. Wu, "Design of rfid active tag system based on msp430," in *IET 2nd International Conference on Wireless, Mobile and Multimedia Networks (ICWMMN 2008)*, 2008, pp. 139–142.

[14] J. Edstrom, D. Chen, Y. Gong, J. Wang, and N. Gong, "Data-pattern enabled self-recovery low-power storage system for big video data," *IEEE Transactions on Big Data*, vol. 5, no. 1, pp. 95–105, 2017.

[15] N. Gong, S. A. Pourbakhsh, X. Chen, X. Wang, D. Chen, and J. Wang, "Spider: Sizing-priority-based application-driven memory for mobile video applications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 9, pp. 2625–2634, 2017.

[16] R. Ge, Z. Lin, N. Gong, and J. Wang, "Design and performance analysis of energy harvesting sensor networks with supercapacitor," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2017, pp. 64–67.

[17] Y.-L. He, S.-Q. Geng, X.-H. Peng, L.-G. Hou, X.-K. Gao, and J.-H. Wang, "Design of outdoor air quality monitoring system based on zigbee wireless sensor network," in *2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, 2016, pp. 368–370.

[18] G. Shu-qin, W. Jin-hui, Z. Lei, W. Wu-chen *et al.*, "A low-power active rfid portable reader system," in *2008 2nd Annual IEEE Systems Conference*. IEEE, 2008, pp. 1–4.

[19] V. C. Emeakaroha, N. Cafferkey, P. Healy, and J. P. Morrison, "A cloud-based iot data gathering and processing platform," in *2015 3rd International Conference on Future Internet of Things and Cloud*, 2015, pp. 50–57.

[20] C.-W. Tseng and C.-H. Huang, "Toward a consistent expression of things on epcglobal architecture framework," in *2014 International Conference on Information Science, Electronics and Electrical Engineering*, vol. 3, 2014, pp. 1619–1623.

[21] R. Hummen, M. Henze, D. Catrein, and K. Wehrle, "A cloud design for user-controlled storage and processing of sensor data," in *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, 2012, pp. 232–240.

[22] J. Fu, Z. Liao, J. Liu, S. C. Smith, and J. Wang, "Memristor-based variation-enabled differentially private learning systems for edge computing in iot," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9672–9682, 2020.

[23] R. Piyare, S. Park, S. Y. Maeng, S. H. Park, S. C. Oh, S. G. Choi, H. S. Choi, and S. R. Lee, "Integrating wireless sensor network into cloud services for real-time data collection," in *2013 International Conference on ICT Convergence (ICTC)*, 2013, pp. 752–756.

[24] J. Fu, Z. Liao, and J. Wang, "Level scaling and pulse regulating to mitigate the impact of the cycle-to-cycle variation in memristor-based edge ai system," *IEEE Transactions on Electron Devices*, vol. 69, no. 4, pp. 1752–1762, 2022.

[25] Z. Liao, J. Fu, and J. Wang, "Ameliorate performance of memristor-based anns in edge computing," *IEEE Transactions on Computers*, vol. 70, no. 8, pp. 1299–1310, 2021.

[26] M. Oli-Uz-Zaman, S. A. Khan, G. Yuan, Y. Wang, Z. Liao, J. Fu, C. Ding, and J. Wang, "Reliability improvement in rram-based dnn for edge computing," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2022, pp. 581–585.

[27] G. Yuan, Z. Liao, X. Ma, Y. Cai, Z. Kong, X. Shen, J. Fu, Z. Li, C. Zhang, H. Peng *et al.*, "Improving dnn fault tolerance using weight pruning and differential crossbar mapping for reram-based edge ai," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2021, pp. 135–141.

[28] J. Fu, Z. Liao, N. Gong, and J. Wang, "Linear optimization for memristive device in neuromorphic hardware," in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2019, pp. 453–458.

[29] S. A. Khan, M. Oli-Uz-Zaman, and J. Wang, "Pawn: Programmed analog weights for non-linearity optimization in memristor-based neuromorphic computing system," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 13, no. 1, pp. 436–444, 2023.

[30] M. Oli-Uz-Zaman, S. A. Khan, G. Yuan, Z. Liao, J. Fu, C. Ding, Y. Wang, and J. Wang, "Mapping transformation enabled high-performance and low-energy memristor-based dnns," *Journal of Low Power Electronics and Applications*, vol. 12, no. 1, p. 10, 2022.

[31] J. Wang, W. Wu, N. Gong, and L. Hou, "Domino gate with modified voltage keeper," in *2010 11th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2010, pp. 443–446.

[32] J. Wang, N. Gong, S. Geng, L. Hou, W. Wu, and L. Dong, "Low power and high performance zipper domino circuits with charge recycle path," in *2008 9th International Conference on Solid-State and Integrated-Circuit Technology*. IEEE, 2008, pp. 2172–2175.

[33] M. V. Moreno, J. Santa, M. A. Zamora, and A. F. Skarmeta, "A holistic iot-based management platform for smart environments," in *2014 IEEE international conference on communications (ICC)*, 2014, pp. 3823–3828.

[34] H. Gupta, D. Bhardwaj, H. Agrawal, V. A. Tikkiwal, and A. Kumar, "An iot based air pollution monitoring system for smart cities," in *2019 IEEE International Conference on Sustainable Energy Technologies and Systems (ICSETS)*, 2019, pp. 173–177.

[35] Y. E. Gelogo, H. J. Hwang, and H.-K. Kim, "Internet of things (iot) framework for u-healthcare system," *International Journal of Smart Home*, vol. 9, no. 11, pp. 323–330, 2015.

[36] E. Luo, M. Z. A. Bhuiyan, G. Wang, M. A. Rahman, J. Wu, and M. Atiquzzaman, "Privacyprotector: Privacy-protected patient data collection in iot-based healthcare systems," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 163–168, 2018.

[37] J. Ma, X. Li, H. Wen, Z. Fu, and L. Zhang, "A key frame extraction method for processing greenhouse vegetables production monitoring video," *Computers and electronics in agriculture*, vol. 111, pp. 92–102, 2015.

[38] A. Botta, W. De Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and internet of things: a survey," *Future generation computer systems*, vol. 56, pp. 684–700, 2016.

[39] N. Pavón-Pulido, J. López-Riquelme, R. Torres, R. Morais, and J. Pastor, "New trends in precision agriculture: A novel cloud-based system for enabling data storage and agricultural task planning and automation," *Precision agriculture*, vol. 18, pp. 1038–1068, 2017.

[40] Juras, "ensorflow-lite-object-detection-on-android-and-raspberry-pi," https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi, 2023, accessed: (Dec. 1, 2023).