An Exploration of Mapping Multimodal Physiological Sensors to Control a Robot Agent

December 14, 2021

Anthony Banks
INFO 698
The University of Arizona
Tucson, AZ, USA

Section 1: Introduction

Technology has significantly advanced over the past decades and the advances are possibly best exemplified by comparing the convoluted analog system used by Doc. Brown to coordinate his alarm clock with starting the morning coffee, toasting bread, turning on the television, and filling his dog's bowl with breakfast in the 1985 blockbuster hit *Back to the Future.* The same process which required multiple analog clocks and a mess of direct cables attaching sensors to machines can be accomplished today using a single centralized device such as a cell phone or a computer. These advances are seen widespread today in part to wireless connections using Bluetooth and Wi-Fi. The Internet-of-Things (IOT) is used to describe the applications of wireless connectivity between several devices, and it has many commercial applications. IOT innovations can also be applied to research and development of new teaching techniques, including the introduction of robotic agents to assist with delivery of curriculum.

The research described herein was conducted to explore novel ways of wirelessly connecting multimodal physiological sensors in near real-time and processing that data to control a robot agent. The architecture was developed to be adaptive of additional sensors and to use the processed data for more general purposes than those specified in the sections below. The code for the system architecture is located at the following public GitHub repository: https://github.com/aibanks/Multimodal-Sensors-to-Control-Robot.

Section 2: Related Work

A software architecture for multimodal sensor integration was described in Gonzales-Sanches et al. (2011). The core components of their architecture include sensing devices, specialist agents, a centre agent, and third-party systems. The sensing devices have a one-way communication to their specialist agent, so the specialist agent is receiving data from the sensing device, but the sensing device is not receiving any data from the specialist agent. Each specialist agent has a two-way communication with the centre agent, so the specialist agent can send data to and receive data from the centre agent. The purposes of the communication from the centre agent to a specialist agent include to coordinate sample rates across multiple devices. Similarly, the centre agent has a two-way communication with third-party systems.

Each specialist agent is uniquely designed around the sensing device. The specialist agent has multiple components which perform the following tasks: communicating with the sensor, preliminary processing or reformatting of the data, sending the data to the centre agent. Some manufacturers offer SDKs and/or APIs for one to communicate with the sensor via Bluetooth, while others use USB ports. The specialist agent is responsible for reformatting the data so that it is sent in the desired format to the centre agent. While there exists a specialist agent for each sensor, the architecture uses one centre agent. The centre agent is responsible for communicating with every specialist agent, processing the multimodal data, and communicating to the third-party software. The way in which the centre agent processes the multimodal data can vary by task, but always integrates the multiple data streams to update the current state. The centre agent communicates the current state to the third-party software at a frequency set within the architecture design.

Gonzales-Sanches et al. describe the challenges presented by vastly different sample rates for each sensor and present some options for overcoming these challenges. One example of overcoming the sampling differences is to record the multimodal data as a data table and continue to add a new row anytime any sensor gets new data; the sensor attributes that received new data will be updated in the new row and any sensor attributes that did not receive new data have their previous data included on the new row. The current state can be calculated at any time based on the newest row in the data table. This solution works best for continuity of data but can create misleading interpretations of the data if one sensor has not been updated for a relatively longer period than other sensors, because the interpretation assumes that the data indicated is accurate for that timestamp. Another solution is to adjust the sample rate from the sensors with the highest sample rate so that each sensor collects one sample every time the slowest sensor collects a sample. Both methods can be used successfully but have drawbacks, so it comes to the application of the data and the sensor-types when determining a solution.

There are multiple techniques one can use to connect a sensor to a computer for data collection and processing. Historically, a direct connection from the sensor to a computer was the predominant method, however, many sensors are becoming equipped with Bluetooth or other wireless connection options. In order to receive the incoming data, a continuous function must be running to keep the connection to the sensor and be ready to receive any data messages. The need for a continuously running function for the connection to the sensor can pose problems for programmers who want to not only receive the data stream, but also process the data and use the processed data to make actions in near-real time. One option is to utilize multi-threading which can run continuous functions separately from other functions within the same Python script, however, communication between the functions when using multithreading can quickly be a complicated process. Another option is to use Message Queuing Telemetry Transport (MQTT) to publish messages from one actively running python script to another. MQTT uses a publish-subscribe model to send messages between devices but can also be used to communicate data between two actively running python scripts. The communication can be unidirectional if one python script is connected as a message publisher and the other is connected as a message subscriber, or it can be bidirectional if both scripts connect as publishers and subscribers. MQTT connections are made using internet connections, so the data from one sensor could be transmitted remotely for processing. MQTT connections are made using a broker, then selecting a channel and topic thread. One python script may publish to and subscribe to multiple channels and topics at once.

The University of Arizona's Robo-Tangible Activities for Geometry (TAG) project is an ongoing research project focused on using robot agents to help teach complicated geometry concepts (Girotto et al, 2016). The TAG project's current application of robotic teaching agents is currently using the LEGO® MINDSTORMS® EV3 robot in combination with a cartesian plane projected onto a white floor mat. The EV3 robot combines a Wi-Fi-capable EV3 Intelligent Brick to two medium motors each connected to a medium wheel on either side of the EV3 Intelligent Brick (**Figure 1**). The TAG project installed a python script onto the EV3 which allows one to send movement commands to the EV3 from a remote computer by communicating using MQTT protocols and leveraging the LEGO® MINDSTORMS® Education EV3 MicroPython SDK (Kumar 2021). The TAG project's EV3 is designed to move upon receiving an MQTT message in the format of: *MoveTank {left speed} {right speed}*, where *{left speed}* and *{right speed}* are floating point values ranging from 0 to 100 and used to denote the percentage of max speed at which the left and right motors should turn the left and right wheels, respectively. For example, the command *MoveTank 50 50* would prompt the TAG EV3 to power both wheels at 50% max power, therefore resulting in a forward movement. Alternatively, the command *MoveTank 50 0* would prompt the EV3 robot to power the left wheel at 50% max power while the right wheel remains stationary, therefore resulting in a spinning motion centered around the right wheel.

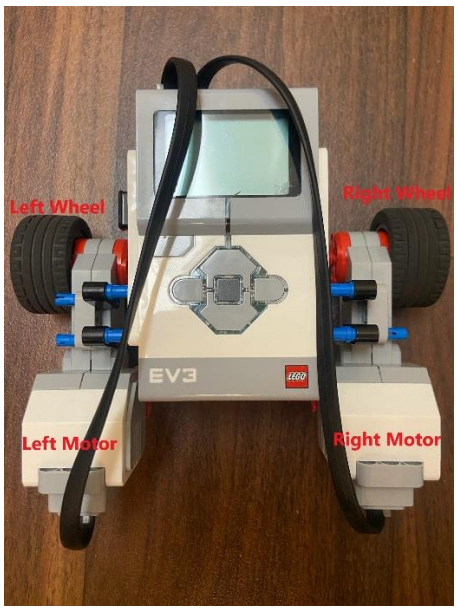

Figure 1. LEGO® MINDSTORMS® EV3 Robot

Section 3: Development of EV3 Robot Movement Commands

The team leveraged the software loaded on the TAG EV3 to receive MQTT messages in the form of *MoveTank {left speed} {right speed}* to move the robot, however, the existing movement commands were limited. The researchers enhanced the existing commands by developing the python script *sender_functions_TB.py* which has a variety of pre-formulated messages to send over MQTT for movement commands such as: go straight, stop, go straight for a preset amount of time, move with bi-speed controls, turn around 180-degrees, spin, and spin for a preset amount of time. This python script was developed to be imported by centre agents and to be used to communicate to the third-party software (EV3 robot) to facilitate additional movement commands.

Section 4: Mapping EEG Signals to Movement Commands

The first sensing device the researchers chose was the EMOTIV Insight EEG brainwave detection sensor ("the Insight"). The Insight is a 5-channel EEG with a 1-to-2-minute preparation time and utilizes a proprietary 2.4GHz wireless BLE connection. The relatively fast preparation time and wireless connectivity made the Insight the ideal EEG for the researchers because it would work best for future deployment with children in a classroom setting if integrated with the TAG project. The Insight can stream the raw EEG data directly from the sensors, but it can also stream Performance Metrics (stress, engagement, interest, relaxation, focus, and excitement). The Performance Metric data for each stream is collected as a floating-point value between 0 and 1, where 0 indicates zero value of that metric and 1 means the highest possible value of that metric (**Figure 2**).

time	Engagement	Excitement	Stress	Relaxation	Interest	Focus
1639518734	0.425095	0.121061	0.22273	0.164797	0.468894	0.237571
1639518744	0.384341	0.113318	0.223685	0.173448	0.471578	0.22749
1639518754	0.401234	0.119259	0.233276	0.188579	0.473602	0.261561
1639518764	0.398204	0.112099	0.239738	0.179768	0.475511	0.25622
1639518774	0.393079	0.111591	0.254822	0.203133	0.478304	0.345361
1639518784	0.341006	0.131152	0.245451	0.202132	0.480463	0.232172
1639518794	0.38812	0.152795	0.259926	0.21001	0.482537	0.19524

Figure 2. Sample EEG Performance Metric Data

The Insight's specialist agent architecture incorporates the Emotiv© Cortex API and an adaptation of the *cortex.py* and *sub_data.py* python scripts from the Emotiv's Cortex example public GitHub repository (Emotiv 2021). The modified python scripts allowed the researchers to subscribe to the Performance Metrics from the Insight, and the data was immediately sent to the centre agent using MQTT protocol.

The researchers experimented with mapping the Insight's data into movement commands for the EV3 robot, the researchers developed the centre agent python script, *int_func_EEG_1d_onMSG.py* to receive the Insight data sent over MQTT from the EEG specialist agent and send a command to the EV3 to move forward, stop, move backward, or stop continuously upon receipt of a message from the EEG specialist agent. The test script worked successfully, so the team built off that script to send movement commands to the EV3 based on the content of the Insight data – not just on the receipt of new data.

The centre agent developed by the researchers to control the movement of the EV3 robot using the Performance Metrics of the Insight EEG is included in the project's GitHub repository as the python script *int_func_EEG_1d_Rel.py*. The mapping of the Insight's Performance Metric data into movement commands for the EV3 robot was utilized by mapping movements in one dimension so the robot moved forwards or backwards during any movements – both wheels were powered to always matching speeds. The speed and direction of the EV3 robot was dictated by the changes in the Insight's relaxation Performance Metric from the previous sample to the newest. The centre agent stored all Performance Metric values received from the MQTT channel from the Insight's specialist agent to calculate the change from sample to sample. If the relaxation metric increased, then the EV3 robot was sent a message to move forward; the amount that the EV3 was told to power its left and right motors was calculated by subtracting the previous relaxation value from the current relaxation value, then multiplying by 100. For example, if the relaxation value was 0.50 and then next value was 0.60, then the

centre agent would calculate the left and right motor power to be a value of 10 – resulting in the EV3 powering its left and right motors at 10% of maximum power.

The mapping successfully worked as intended, however the researchers quickly noticed that the Insight's Performance Metrics are difficult to control even if trying deliberately, so the mapping was not very successful as it relates to the participant consciously controlling the speed of the EV3 robot in a one-dimensional case, let alone in a two-dimensional case where there are movement commands for the speed and direction of the robot. However, the researchers determined that the Performance Metrics would likely work well when actively being collected, but the centre agent only triggering movement commands when a metric exceeds a range instead of actively mapping the changes from sample to sample.

Therefore, the researchers sought to bring in another sensor which could compliment the Insight's Performance Metrics with sensory data that has a participant can tangibly control with little training, so they could use the multimodal sensors in unison to control the EV3 robot.

Section 5: Mapping Empatica E4 Signals to Movement Commands

The second sensor the team integrated into the EV3 movement architecture was the Empatica E4 wristband (**Figure 3**). The E4 wristband is a medical-grade wearable device that offers real-time physiological data acquisition. Equipped with sensors to monitor blood volume pulse, galvanic skin response, 3-axis accelerometer, and skin temperature, the E4 sensor was an ideal choice for the variety of data streams accessible from the single device and therefore vast variety of applications and movement mapping options.

The E4 can connect to a computer using a BLE-Bluetooth dongle and is worn similar to a standard watch without any nuisances to the wearer. Empatica offers an API for developers to stream the data for their own purposes. In order to do so, one must acquire an API key from Empatica, install the Empatica Streaming Server software, and connect to the streaming server using Transmission Control Protocol/Internet Protocol (TCP/IP). The researchers' desire for a wireless sensor with quick set-up time and mild to no nuisances to the wearer, multiple sensors and data streams, and high sample rate made the Empatica E4 wristband a desirable sensor to integrate to their architecture.

Figure 3. Empatica E4 Wristband

The research team built off existing research from the University of Arizona's SensorLab for streaming E4 data using in a python script but modified to send the data to a centre agent over MQTT (see python script *sub_data_E4.py* from the project GitHub repository). The python script connects to the streaming server with TCP/IP, subscribes to the pre-specified data streams, and receives data message text strings containing data from multiple timestamps and data streams from the streaming service at regular intervals. Additionally, the script transforms each data message text string into a list of dictionary objects with one dictionary object per unique data stream and timestamp and publishes the list of dictionary objects over MQTT to the centre object.

To compliment the Insight EEG's slower sampling rate and difficult-to-change metrics, the researchers experimented with mapping the E4's accelerometer data stream to the EV3 robot's movement controls. "The E4 is equipped with a MEMS type 3-axis accelerometer that measures continuous gravitational force (g) applied to each of the three spatial dimensions (x, y, and z). The scale is limited to +/-2g." (Empatica 2021). The raw accelerometer data from the E4 is reported as 1/64g, so the raw data values are within the range of -128 to 128. The accelerometers can detect movements and will report those movements relative to the spatial dimensions. Additionally, the accelerometers detect the constant force of the Earth's gravity, 1g, and the orientation of the wristband dictates the extent to which each spatial dimension detects the components of the background acceleration due to the Earth's gravity. Figure X.2 illustrates the orientation of the E4 wristband at which the entirety of the background acceleration from Earth's gravity ("background acceleration") is registered in one spatial dimension while the other dimensions register 0g.

The researchers mapped the E4's accelerometer data to the EV3 robot's movement controls by leveraging the background acceleration recorded for each spatial dimension for different orientations of the E4 wristband. The specialist agent performed the tasks from the python script *sub_data_E4.py* as described above, while subscribing to the accelerometer data. The centre agent subscribed to the MQTT messages from the specialist agent to receive the data as a list of dictionary objects for each sample point. Due to the high sample rate, the centre agent used the last datapoint from every message received to process the movement controls. Similarly, the rate at which the specialist agent published to MQTT was controlled and could be easily adjusted for faster or slower response times.

The centre agent for the EV3 movement control mapping relative to the E4 accelerometer data as described above is included in the project GitHub repository as *int_func_E4_3d_Acc.py*. The centre agent processes the spatial acceleration data and maps the data to movement controls for the EV3 robot using the x and y spatial dimension acceleration data primarily as it's derived from background acceleration. The acceleration in the y spatial dimension was mapped to control the speed of the EV3, and the acceleration in the x spatial dimension was mapped to control the off-axis acceleration, or direction and extent to turn, for the EV3. When the y spatial dimension's acceleration is 0, then the EV3 will be stopped. Furthermore, a positive value for the y spatial dimension's acceleration results in the EV3 powering its wheels forward, and a negative value results in powering its wheels backward. The y spatial dimension controls the maximum power for either of the EV3 motors at any given time, and at least one of the EV3 motors will be powered at that maximum at any given time.

The x spatial dimension reduces the power for one of the wheels while the other is powered at the relative maximum, therefor resulting in off axis acceleration which turns the EV3 to the left or right. When the x spatial dimension's acceleration is 0, then both wheels will be powered at the maximum

power dictated by the acceleration in the y spatial dimension and therefor the EV3 will move in a straight line. However, if the x spatial dimension's acceleration is less than 0, then the EV3 will power its right wheels at the relative maximum power dictated by the y acceleration and reduce the power to its left wheels, therefore resulting in the EV3 to turn to the left. Similarly, if the x spatial dimension's acceleration is greater than 0, then the EV3 will power its left wheels at the relative maximum power and reduce the power to its right wheel, therefore resulting in the EV3 to turn to the right.

When the E4 wristband is worn properly, the background acceleration registered in the y spatial dimension of the E4's accelerometer, and therefore the relative maximum power of the motors, increases when the wearer's wrist is lower than their elbow; the maximum background acceleration in the y spatial dimension is registered when the wrist is directly below the elbow. Similarly, the background acceleration registered in the y spatial dimension of the E4's accelerometer becomes negative when the wearer's wrist is above their elbow; the most negative background acceleration in the y spatial dimension is registered when the wearer's wrist is directly above their elbow. Alternatively, when the E4 wristband is worn properly on the participant's left arm, the background acceleration registered in the x spatial dimension of the E4's accelerometer increases when the arm is rotated clockwise – the user's left thumb is lower in elevation than their left pinky finger – and therefore results in the EV3 robot to power its left wheels at the relative maximum power while powering its right wheels at reduced power which leads the EV3 to turn to the right. Furthermore, when the E4 wristband is worn properly on the participant's left arm, the background acceleration detected in the x spatial dimension of the E4's accelerometer becomes negative when the participant's left forearm is rotated counterclockwise – so that their left thumb is at a higher elevation than their left pinky finger – and therefore results in the EV3 robot to power its right wheels at the relative maximum power while powering its left wheels at a reduced power which leads the EV3 to turn to the left. The EV3 will turn the most to the left or right when the participant's forearm is rotated so that their left thumb is directly above their left pinky finger or so that their left thumb is directly below their left pinky finger, respectively.

The centre agent for the EV3 movement control mapping relative to the E4 accelerometer data uses the process described above to calculate the power for the EV3 robot's left and right motors for each data message received from the MQTT channel with the E4's specialist agent. Upon calculating the power for the left and right motors, uses the *move_bispeed* function from the imported python script *sender_functions_TB.py* which sends a message to the EV3 robots MQTT channel – different channel than the one from the specialist agent to the centre agent – in the form of *MoveTank {left speed} {right speed}* using the left and right speeds calculated from the process described above (**Figure 4**).

The research team found that the mapping procedure did not result in the expected movement commands when the wristband was oriented in a position where both the x and y spatial dimensions would register non-zero values – such as a position where the participant's wrist was at a 45-degree angle below their elbow and the forearm was semi-rotated clockwise. The described orientation results in positive x and y values, which maps to a forward direction with a turn to the right. However, because the accelerometers are mostly leveraging the background acceleration, 1g, the speed is less than expected for a similar orientation of the wrist below the elbow, but without the forearm rotation. The mapping is more intuitive if the rotation of the forearm does not significantly reduce the speed registered from a similar orientation in the y spatial dimension.

Time (UNIX)	Acc_x (1/64g)	Acc_y (1/64g)	Acc_z (1/64g)	Left_Speed	Right_Speed	Comments
1639348385	-7	2	63	4	4	Low Speed
1639348386	-6	18	61	13	14	Increasing Speed
1639348387	-1	20	56	16	16	Moving Straight
1639348388	-1	31	53	24	24	Moving Straight
1639348389	-3	24	57	19	19	Moving Straight
1639348390	4	31	56	24	23	Moving Straight
1639348391	1	36	54	28	28	Moving Straight
1639348392	-2	56	20	43	44	Moving Straight
1639348393	-2	59	16	45	46	Moving Straight
1639348394	-3	59	12	45	46	Moving Straight
1639348395	-2	59	12	45	46	Moving Straight
1639348396	-1	59	11	46	46	Moving Straight
1639348397	28	53	5	41	32	Turning Right
1639348398	35	52	2	41	30	Turning Right
1639348399	46	38	4	33	21	Turning Right
1639348400	32	49	8	38	28	Turning Right
1639348401	22	56	8	44	36	Turning Right
1639348402	-25	56	8	35	44	Turning Left
1639348403	-33	51	10	30	40	Turning Left

Figure 4. Annotated E4 Accelerometer Data with Corresponding Left and Right Motor Power

To overcome the issue described above, the researchers implemented additional procedures in centre agent. Specifically, if the absolute value of the x spatial dimension's acceleration exceeded the absolute value of the y spatial dimension's acceleration, then the y spatial dimension's acceleration as used for the calculation of the relative maximum motor power was increased in absolute value to the mean of the x and y spatial dimension's acceleration, while preserving the sign, positive or negative, of the original y dimensions acceleration. This allowed the EV3 to conduct significant turns with enough speed to carry out the maneuver in a responsive manner.

After implementing the procedures described above, the E4 accelerometer signals were successfully mapped to movement commands for the EV3 robot which could be intuitively used by a participant and had a fast response rate.

Section 6: Mapping Multimodal Sensors to Movement Commands

The researchers integrated the above-described processes to map Insight EEG signals and Empatica E4 signals to movement commands for the EV3 robot. The integration process was conducted entirely in the centre agent and therefore did not require any alterations to the specialist agents.

The specialist agents published their data on unique MQTT channel topics, so the centre agent subscribed to both channels. However, when a message was received by either topic, the centre agent triggered the same *on-message* function, which required significant updating to integrate the steps required for EEG or E4 data. The messages from both specialist agents were composed of python dictionary objects, so the first step in the integrated on-message function is to look at the keys of the dictionary, because that gives away whether the message is formatted as EEG or E4 data. Upon receipt of a new MQTT message, the centre agent updates the *message_source* variable – to either "EEG" or "E4" – which is used for additional steps in the data processing, storage, and ultimately determining the movement command to send to the EV3 robot.

The researchers integrated the main components of the original EEG and E4 centre agents into the multimodal sensor centre agent, python file *int_func_multimodal.py*, and used the message_source variable to trigger the data storage and movement commands related to the two sensors. The data processing and movement command mapping from the E4 centre agent remained consistent for the E4 components of the multimodal sensor centre agent, so the E4 could provide continuous left, right, forward, and backwards movement commands to the EV3 robot based on the x and y spatial dimension's acceleration readings. Alternatively, the researchers developed a new mapping for the EEG data to be used in the multimodal sensor centre agent as opposed to the original EEG centre agent discussed above. The new mapping for the EEG Performance Metrics was designed to continuously monitor the metrics but only trigger a movement command when a value was outside a preset range of values. For example, the relaxation Performance Metric was mapped to send no movement commands when its values are within the range of 0.4 to 0.6, but to send movement commands for the EV3 robot to spin at 80% max power for a duration of two seconds counterclockwise or clockwise if the value is below or above the range of values, respectively.

The multimodal sensor centre agent described above successfully combined the most successful components of the individual sensor centre agents described above. The multimodal sensor movement mapping architecture would predominantly send movement commands to the EV3 based on the current orientation of the E4 wristband, but those commands would be overwritten and the EV3 robot would be told to spin whenever the EEG Performance Metrics fell outside the desired range. The multimodal architecture was tested using the accelerometer data from the E4 wristband and the relaxation Performance Metric from the Insight EEG, however it was developed to be generalized so it could be adapted to include movement mappings based on the other E4 data streams - blood volume pulse, galvanic skin response, and skin temperature – and/or the other Insight EEG Performance Metrics – engagement, excitement, long term excitement, stress/frustration, interest/affinity, and focus. Additionally, the multimodal architecture was developed in such a way that additional sensors can be added for additional inputs and movement mappings without disruption as long as the additional sensors follow the same specialist agent processes described above. In such a case, the centre agent would need to be updated to subscribe to the additional data stream, the on-message function would need to be able to identify messages from the additional sensor's data stream, and the additional sensor's data would require a movement mapping.

The multimodal sensor movement mapping architecture described in this section mapped the EEG and E4 data to movement controls without accounting for the data from the other data stream; for example, if the EEG data was within the acceptable range, then the E4 data would entirely dictate the movement of the EV3 robot, and when the EEG data was outside the acceptable range, then the E4 data would not be used to control the EV3 robot for two seconds. Future research could further develop the movement mapping schema so that both sensors are always contributing to the EV3 robot's controls, such as the speed being dictated by one of the EEG's Performance Metrics while the direction is dictated by the E4's accelerometer data.

Section 7: Conclusions

The research successfully explored the ways in which biophysiological sensor data could be mapped to movement controls for a LEGO® MINDSTORMS® EV3 robot. In doing so, a data collection, aggregation, and processing architecture was developed using a framework of specialist agents, centre agent, and

third-party software as described by Gonzales-Sanches et al. The research tested strategies of mapping one sensor to movement controls, as well as using multimodal sensors. The mapping of EEG Performance Metric data to movement controls was found to be difficult to use tangibly to control the robot in a desired way, even in a 1-dimensional field of play. The researchers found that the Empatica E4's accelerometer data, however, could be mapped in a way that was intuitive for participants to understand. Additionally, the researchers found that the EEG Performance Metric data could be successfully mapped to movement controls when used to perform a limited quantity of movement commands all of which are triggered by the data, while the majority of the movement commands were controlled by another source, such as the E4's accelerometer.

The framework developed by the researchers can be applied more generally than the uses described herein. The framework was tested using a subset of the total data streams from the two sensors, so future research can be conducted by adapting the specialist and centre agents to collect additional data streams and mapping those data streams to movement commands. Similarly, the research can be further developed by exploring the mappings from a combination of the multimodal sensor data, such as controlling the speed of the robot from the EEG's Performance Metrics while controlling the direction – left, right, forward, backward – of the robot from the E4's accelerometer data.

The framework could also be adapted and applied to the University of Arizona's TAG project. The TAG project will be using pre-scripted movement commands to drive the EV3 robot across the floor with a cartesian grid projected upon it to draw geometric shapes. The pre-scripted movement commands to draw the geometric shapes could act as a specialist agent feeding data into the centre agent which in turn communicates to the EV3 robot for the movement commands. The TAG project could integrate the EEG data mapping process to alter the pre-scripted movement commands so the EV3 robot moves differently based on one or more Performance Metrics measured by the EEG. For example, if the excitement Performance Metric falls below a threshold, therefore insinuating the student is not very interested in what's happening, then the centre agent could speed up the EV3 to make its movements across the floor quicker and more exciting. Alternatively, it could add a high-speed spinning movement command that makes multiple spins before continuing to drive and mark the outline of a shape every time the robot would be changing directions.

The research opened the door to additional exploration of multimodal sensory mapping for robotic movement controls, as well as multimodal sensory mapping to other third-party software. The architecture was developed to be adopted for additional sensors, data streams, data processing methods, and third-party software including but not limited to robotic agents.

Section 8: Works Cited

Emotiv. (2021). "cortex-v2-example." GitHub repository, https://github.com/Emotiv/cortex-v2-example/tree/master/python

Empatica. (2021). "How is the acceleration data formatted in E4 connect?" https://support.empatica.com/hc/en-us/articles/202028739-How-is-the-acceleration-data-formatted-in-E4-connect-

Girotto, Victor & Lozano, Cecil & Muldner, Kasia & Burleson, Winslow & Walker, Erin. (2016). "Lessons Learned from In-School Use of rTAG: A Robo-Tangible Learning Environment." *CHI'16*, 919-930.

Gonzalez-Sanchez, Javier & Chavez-Echeagaray, Maria-Elena & Atkinson, Robert & Burleson, Winslow. (2011). “ABE: An Agent-Based Software Architecture for A Multimodal Emotion Recognition Framework.” 10.1109/WICSA.2011.32.

Kumar, Rishav. (2021). “MQTTws-ev3.” GitLab repository, https://gitlab.com/rishavk1/mqttws-ev3