Gesture Detection in Insect Farm

Aditi Mishra, Arda Ntourali, Bianca Chiusano, Imanol Mugarza, Simonas Bubnys
Department of Advanced Computing Sciences

Maastricht University

Maastricht, The Netherlands

January 25, 2023

Abstract

This paper aims to outline ways in which gesture detection could be integrated within an intelligent insect farm. Implemented custom gestures will act as a medium for controlling three other aspects of the insect farm: A Robot Arm, AGVs and a programming interface. The paper discusses both the physical setup and the implementation. The first includes the position of the cameras, monitor (running the code) and projector. While the implementation was brought out by the use of tools, mainly the python package CVZone, which uses MediaPipe and OpenCV. A webhook is used as a method to communicate with other parts of the insect farm, while a gesture manual and a sound system (implemented) is provided to communicate with users performing gestures. Overall, a gesture is recognized correctly right away 63.3% of the time. Finally, some experiments such as testing the accuracy of the gestures and measuring the Emotions of users were performed to identify the best possible gestures and setup.

Index Terms **:- Gesture Detection, MediaPipe, CVZone, OpenCV, Insect Farm.**

1 Introduction

The European project CoRoSect, is working on creating intelligent insect farms to attain sustainable food [1]. The purpose of this paper is to help and simulate the CoRoSect project by generating a miniaturised autonomous insect farm. This paper serves to explain the workings and results of Group 8 (Gesture Detection), which is one of the seven groups involved in this project.

The insect farm has many people as visitors for the field study, including small kids. To non-programmers, it is not intuitive how technical components in the farm function. To facilitate this, the use of gestures makes it easier for non-programmers to follow. To further enhance their experience, they would also be given the freedom to draw a path for an AGV to move in a certain direction. Similarly, they can also gesture with their hands to move the robot arm to pick or drop something.

From this the problem statement can be formally explained as: There is a lack of exposure to complex robotics systems that young pupils can easily interact with. Thus, creating a simple interface, that young inexperienced programmers can easily utilize, to manipulate and control complicated robots and AGVs is important. Our primary focus will be hand and limb gesture detection and the implementation of custom gestures.

The primary research questions our group will focus on are "How can we quantitatively assess the accuracy of the detection of our custom gestures?" and "What qualitative observations can be drawn to ensure correct identification of our custom gestures?".

The article will briefly explain the methodologies (Section 2) which includes the information on setup (Section 2.1), implementation of the gestures (Section 2.2), the API which is used for communication and interaction (Section 2.3) and the Gestures Manual (Section 2.4). The article also describes the experiments performed (Section 3), the Discussion (Section 4) and the conclusion of the whole article (Section 5)[2].

2 Methodology

2.1 Equipment & Setup

For the physical setup, four cameras are used and connected to a projector to enable users to view both themselves on the screen, as well as the actual ant farm setup (Figure 1). Three cameras are used to detect gestures. The first camera is for gestures controlling the robot arm, the second for the AGV and finally, the third camera is for the programming interface modules. The fourth camera is used to simply display an alternative view of the setup. Four cameras were chosen to make full use of the rectangular screen of the projector.

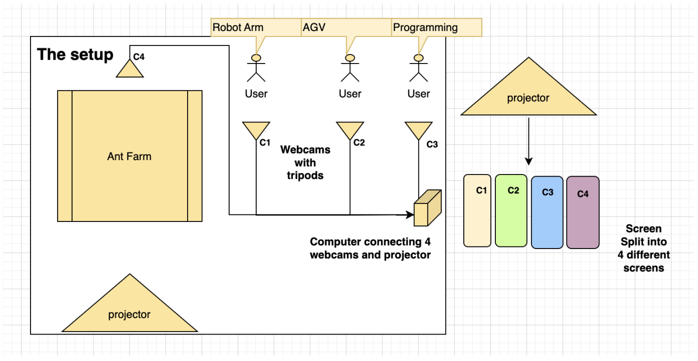


Figure 1: The set up

2.2 Implementation

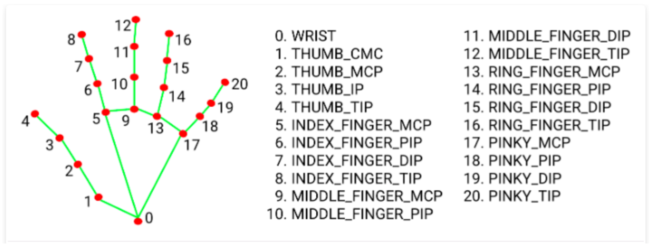
In this project, gesture detection acts as a medium for easy interaction with various parts of the ant farm. As of the project's current state, specific and simple custom gestures were implemented to control the following three modes: the Robot Arm, The AGVs and finally interacting with a friendly programming interface. As previously stated in Section 2.1, three cameras will be set up to film users willing to participate, three cameras for three modes.

2.2.1 MediaPipe and CVZone

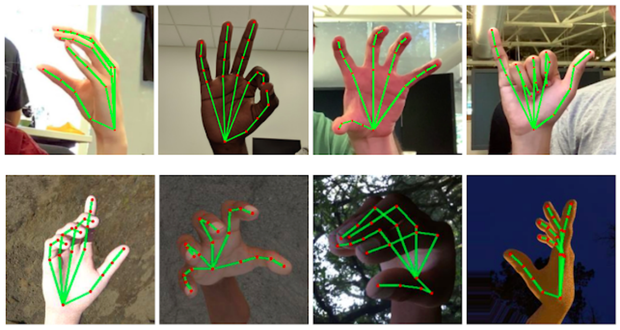
The underlying Python package used for the implementation of the custom gestures is called CVZone. CVZone is a Computer vision package that uses OpenCV and Mediapipe libraries. It is important to first understand the idea behind the working of MediaPipe.

Mediapipe is a framework that offers customizable ML solutions for processing time-series data such as videos and audio. For the implementation of custom gestures, "MediaPipe Hands" is used. The latter is a high-fidelity hand and finger tracking solution, which not only is able to recognize 3D landmarks of a hand from a single frame, but also does not necessarily need a powerful desktop environment to be able to run, which makes it more accessible for students to operate [3].

The Machine Learning Pipeline works with a palm detection model and a hand landmark model [4]. The palm detection module inputs frames of video as images and outputs a box that encapsulates the palm. Using a single-shot detector it is able to correctly bound the palms using 95.7% accuracy. The hand landmark model provides 21 hand landmarks from the wrist to each fingertip [3].



This model was trained by data from around 30 thousand hand images of different possible hand poses which were manually annotated with 21 three-dimensional coordinates [5].



Specifically for this project, cv2 was used for the display, writing, reading and general manipulation of images/videos. In contrast, the HandDetector from the CVZone Hand tracking module was used for hand landmarks recognition [5].

2.2.2 Robotic Arm Control

The first mode is the Robot Arm control. The user might want to take control of the Robot arm, and perform certain actions such as moving or replacing both the crates and/or the AGVs. Custom gestures are used as a more efficient and intuitive way of performing such actions in a non-physical or coding way.

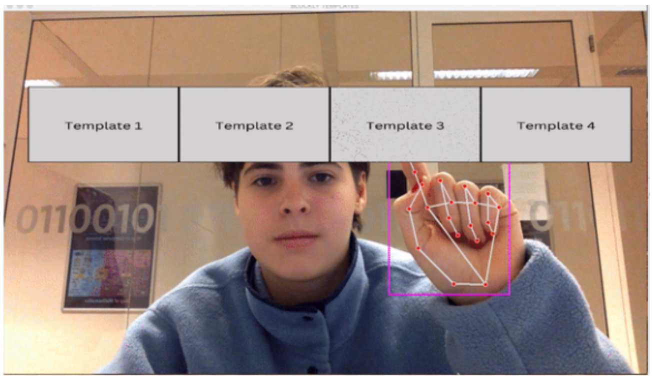
Six custom gestures were implemented for the control of the Robot Arm: Pickup, Drop, Move Right, Move Left, Move Forward and Move Backwards. These gestures trigger the robot arm to perform the actions. The implementation of all these gestures can be found in section 2.4. By performing a pickup gesture, the robot arm gets the indication to pick up a crate. The drop gesture allows the robot arm to drop a crate that it is carrying. The move right and move left gestures allow the robot arm to move right and left respectively while carrying a crate. Lastly, the move forward and move backwards gesture allow the robot arm to move either forward or backwards while carrying a crate.

2.2.3 AGV Control

Moving on to the control of the AGV, the second mode, two approaches were considered namely Path Drawing and simple Custom Gestures.

The first approach, which is path drawing, allows the users to draw a path starting from one location and ending at another location on a custom-generated map of the insect farm. The AGV receives and moves along this path. Thus users are able to have an interactive session with the AGV by moving it to the desired points.

The second approach is to generate simple custom gestures which then trigger the AGV to perform certain actions. The gestures that we have for this are: Move Right, Move Left, Move Forwards and Move Backwards. As explained in the Robot Arm mode above, these actions are similar to each other. Gesturing the AGV to move right or left makes the AGV move towards the right or left. Similarly, gesturing the AGV to move forwards or backward makes the AGV move forward or backwards. While performing these actions, the AGV could also be carrying crates to transport them to another location. Both the Robot Arm control gestures and the AGV gestures are represented in the code using arrays of length 5 for each gesture containing 0s and 1s.



The length of the array is 5 for the 5 fingers of one hand, while 1s stand for lifted fingers and 0s for "closed" fingers. For example, an array [1,0,0,0,0] represents the forward gesture in which only the thumb is shown thus in the array, 1 denotes the thumb and the rest of the fingers are 0 as they are not shown.

2.2.4 Programming Interface Control

Lastly, the third mode consists of gestures implemented to easily interact with a programming interface. On the screen, the user will be able to see that there are some templates of ready-to-run code (Figure 2.2.4). This consists of "Blockly" Code, which is an easy editor that "represents coding concepts as interlocking blocks" [6].

At this point they will be able to perform three custom gestures, one is to select a template, the second gesture can be used to go back and choose another template, and finally, the third gesture can be performed to run the code. In this way, users will still be able to interact with the farm by performing gestures, as well as familiarise themselves with simple Blockly code.

For this prototype, the CVZone Hand Tracking Module was used for keeping track of the position of the user's index finger, in order to check if a certain template, provided by the programming interface group, had been selected. Please consider the following pseudo-code as an example of how the detection takes place:

```
selectedTemplate is template1
detector is HandDetector from CVZone
image from cap.read()
flip image
while True:
    hands from findHands() from detector
    if hands showing:
        lmList is hands[0]
        indexFinger is lmList[8]
        if index[1] > y1 and index[1] <y2:
            if index[0] > x1 and index[0] < x2:
                selectedTemplate is template2
```

When the program starts the first template is already selected. This was done so that the user can understand from the beginning the difference between a selected and non-selected template. For this procedure, the hand detector should only be considering one hand. If a hand is detected, a list of all 21 landmarks is created, this is called lmList. The index finger's XY-coordinates are at position 8 of the list, as it is the index fingertip is 8th landmark. As lmList is a list of coordinates, it can be thought of as a list of lists. Hence, the index's x-coordinate is index[0] and the index's y-coordinate is index[1]. These positions can be used to check when the user's index finger is hovering on top of another template, by checking the position of both the templates and the finger. If that is the case, the wanted template will be selected. Once the user is sure of the template they selected they can confirm by making a thumbs-up gesture and the code of the selected template will be shown. At this point, the user may go back to the templates view if they want to change their selection, or choose to run the code of the selected template. A detailed view of these custom gestures can be found in the Manual (Section 2.5).

In addition to these different modes, gestures also have a sound associated with them. Each time the user performs a gesture, a sound stating the gesture is generated. This makes it easier to perform a gesture as the user keeps getting real-time feedback of the gestures they performed.

2.3 Communication & Interaction: API

The application programming interface for the project makes use of REST principles. The provided API endpoints allow any third party application to manipulate various variables, subscribe a webhook, add gestures, and retrieve a list of gestures.

Since the current status of inter-group integration does not address security concerns, there is no authentication to use these API endpoints.

GET /**gesture**
retrieves a list of available gestures

Response application/json

200 OK

```
[
    {
        "id": 1234567890,
        "name" : "The Gesture",
        "description" : "The
            Description.",
        "metadata" : {
            "some-meta" : "value",
            ...
        }
    }...
]
```

POST **/gesture**
create a new gesture

Parameter

name	name of the new gesture
handMatrix	An array of 5 integers representing each finger being up or down

Response application/json

200 OK

```
[
    {
        "id": 1234567891,
        "name" : "New Gesture",
        "description" : "New
           Description.",
        "metadata" : {
            "some-meta" : "value",
            ...
        }
    }...
]
```

400 Bad Request

```
[
    {
        "message": "Following fields
           cannot be empty for
           creation of a new gesture
           : name, detectionMatrix"
    },
    {
        "message": "A detection
           matrix should not include
            any other values than 0
           or 1"
    },
    {
        "message": "A detection
           matrix should always have
            a length of 5,
           corresponding to each
           finger."
    }
]
```

409 Conflict

```
[
    {
        "message": "Provided gesture
           detection matrix is
           occupied by another
           gesture"
    },
    {
        "message": "Provided gesture
           name is occupied by
           another gesture"
    }
]
```

2.3.1 Webhooks

A webhook is a method for one application to provide real-time data to other applications. It enables one application to send a message, or "event," to another over the internet. In our context, the event is triggered with a gesture being detected, and the message contains the meta-data related to the gesture that was detected. [7]

When a gesture detection event occurs, our application sends an HTTP POST request to all the subscribed webhook endpoints, which the endpoints must be properly configured to receive JSON gesture meta-data by the recipient service. The receiving application, such as the programming interface application, then processes the event and takes appropriate action, such as triggering a template.

Webhooks are useful because they allow for real-time communication between systems without requiring the receiving system to poll the sending system for updates on a regular basis. This can save resources while also improving performance.

Our application programming interface provides endpoints to check status of a webhook, subsribe a new webhook endpoint, and disable/enable an existing webhook.

GET **/webhook**
retrieves the status of a certain subscription

Parameter

address	URL of the webhook endpoint

Response application/json

200 OK

```
{
    "id": 2234567890,
    "callback" : "URL of endpoint",
    "active" : true
}
```

400 Bad Request

```
{
    "message": "No address was
       provided. Provide callback
       address using a query
       parameter (including http/
       https), such as\\API_ENDPOINT
       /webhook?address=http://
       AddressToMyEndpoint.com"
}
```

404 Not Found

```
{
    "message": "Unknown webhook
       address. Add your webhook
       using a POST request to this
       endpoint."
}
```

POST **/webhook**
subscribe a new webhook

Parameter

address URL of the webhook endpoint

Response application/json

200 OK

```
{
    "id": 2234567891,
    "callback" : "New URL of endpoint
        ",
    "active" : true
}
```

400 Bad Request

```
{
    "message": "No address were
        provided. Provide 'address'
        in request body."
}
```

DELETE **/webhook**
disable a webhook

Parameter

address URL of the webhook endpoint

Response application/json

202 Accepted

```
{
    "id": 2234567890,
    "callback" : "URL of endpoint",
    "active" : false
}
```

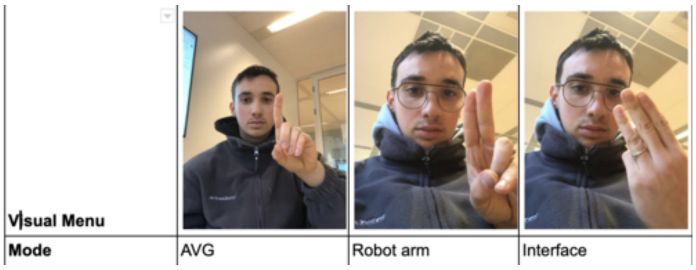
400 Bad Request

```
{
    "message": "Unknown webhook
        address. Add your webhook
        using a POST request to this
        endpoint to disable then."
}
```

2.4 Gestures Manual

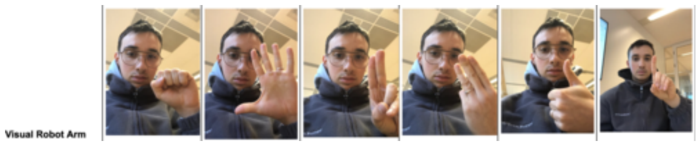
For all modes that this project has, there is an assigned gesture. Once the corresponding gesture has been made, the chosen mode can be used. First mode corresponds to the AGV, second mode to the Robot Arm and the last mode to the Blocky Interface.

- The first mode is assigned, making the index finger gesture.
- The second mode is assigned, making the gesture of the index finger + middle finger.
- The last mode is assigned, making the gesture of the index finger + middle finger + ring finger.



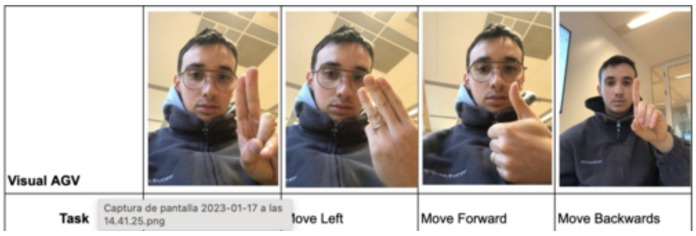
Inside the Robot Arm Mode, in total there are 6 different gestures. The gestures are the following: pickup, drop, move right, move left, move forward and move backwards.

- Pickup gesture is assigned with the closed fist.
- Drop gesture is assigned with the open hand.
- Moving the right gesture is assigned with the index finger + middle finger.
- Moving to the left is assigned with the index finger + middle finger + ring finger.
- Moving forward is assigned with the thumb finger.
- Moving backwards is assigned with the index finger.



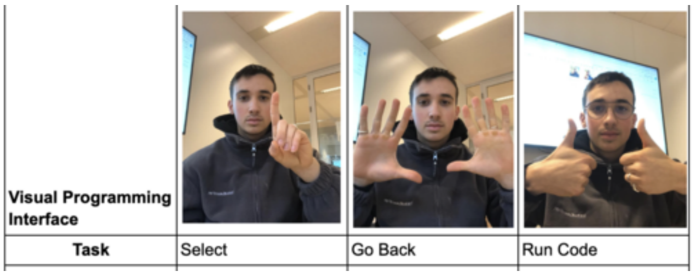
In AGV mode, 4 gestures can be done in this mode: right, left, forward and backwards.

- Moving the right gesture is assigned with the index finger + middle finger.
- Moving to the left is assigned with the index finger + middle finger + ring finger.
- Moving forward is assigned with the thumb finger.
- Moving backwards is assigned with the index finger.



Finally, for the Blockly Interface mode, 3 different gestures can be selected: select, go back and run code.

- Select gesture is done with the index finger.
- Going back is selected with two open hands.
- Running the code is done with two thumbs fingers.



3 Experiments

3.1 Gesture Performance Analysis

The first set of experiments were performed on users having a technical background and on the authors of this report. To test the performance of the gestures, the users were given an array of length 5 containing 0s and 1s. The 0s represented the fingers that were not visible or shown whereas the 1s represented the fingers that were shown. These gestures were randomly generated for each user and the users had to replicate the gestures represented in the array in the correct order. The success rate of this experiment was determined if the user correctly performed the gestures as presented in the list. If the user failed to provide the gesture in the correct order then that counted to the number of fails and if the user performed only 2 out of the 20 gestures present in the list then the remaining 18 gestures that the user did not perform were counted as omitted.

3.2 Questionnaire

The gestures were tested on the basis of their accuracy. Furthermore, a questionnaire of intuitiveness and some questions on their emotions before and after performing the gestures were given to the users to gain insights about the users and their emotions during the whole experiment. There were 12 users in total that were asked to perform these experiments. The users are of different age groups, 16.7% of the users are in the age group 0-19, 41.7% of the users are in the age group 20-30, 33.3% of the users are in the age group 50-69 whereas 8.3% of the users are in the age group 70-100. The following are some analysis based on several questions that were asked to the users purely on the working of the gestures:

- Out of the 12 users, 75% of them are students.
- 58.3% of the users found the predefined gestures intuitive.
- Only 8.3% of the users have a programming background.
- Only 8.3% of the users have used gestures for controlling something in their lives.
- Lastly, half of the users think that the new gestures are intuitive.

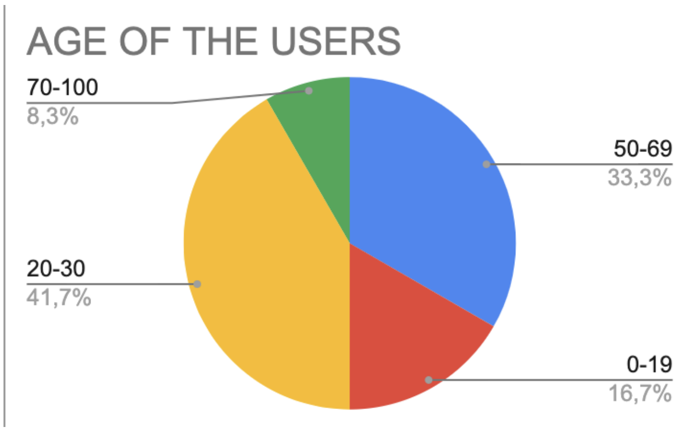


Figure 2: Age survey

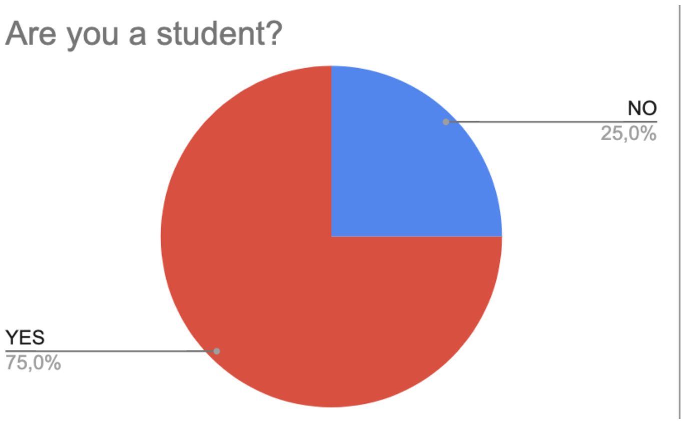


Figure 3: Student survey

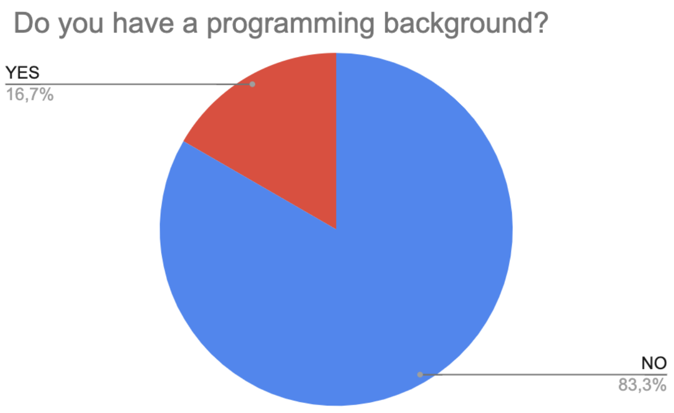


Figure 4: Programming survey

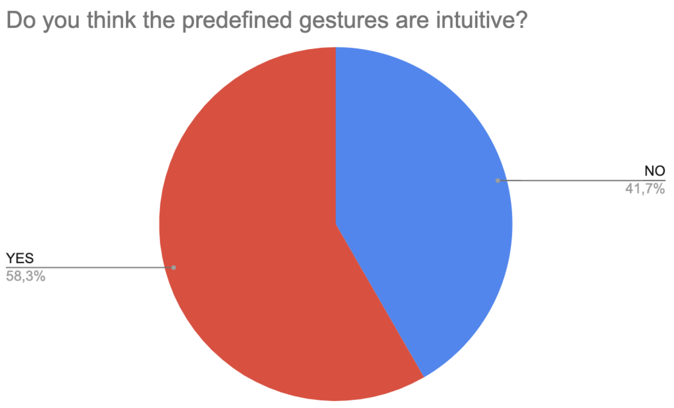


Figure 5: Predefined gestures survey

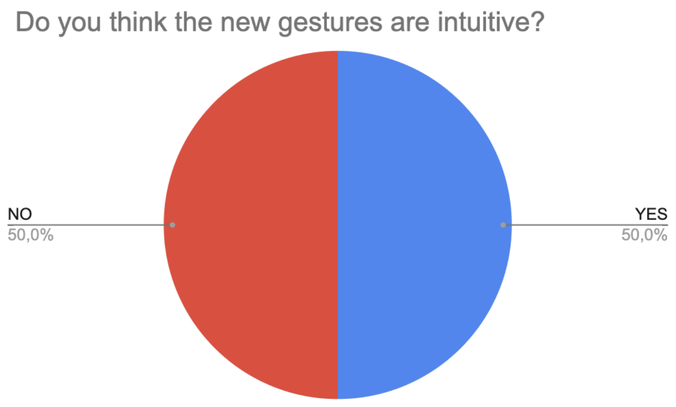


Figure 6: New gestures survey

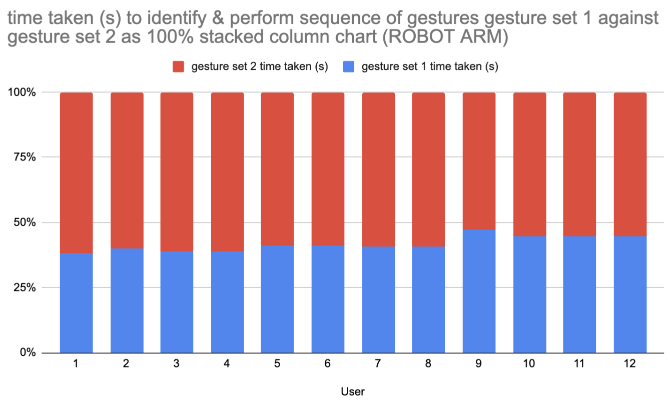


Figure 7: Robot arm accuracy test

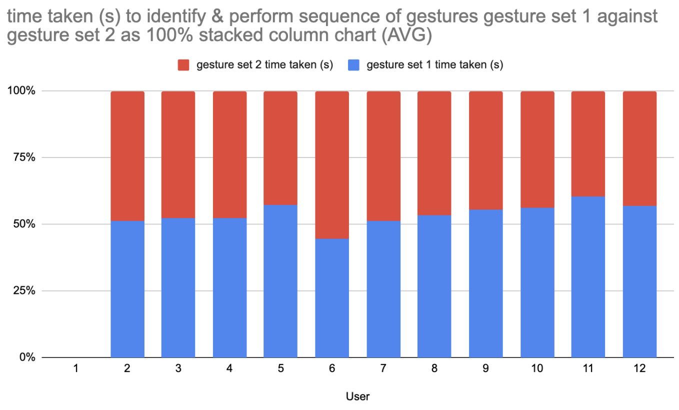


Figure 8: AVG accuracy test

3.3 Accuracy Test

To test the rate of correct gesture detection an experiment was conducted in which the users had to show the gesture asked for on the screen. All gestures were provided and given in a random order. The amount of time the gesture is correctly and incorrectly identified was recorded. Then the success rate of the detected gestures was found and quantitative conclusions were drawn. The goal was to identify gestures with a correctness rate of 90%. Two sets of gestures were provided for each of the three modes. For the robot arm, the first set of gestures were pickup, drop, right, left, forward, backward; and the second set of gestures were pickup, drop, right, left, forward and backward. The difference between the first and second set of gestures is in the left and right gestures in each of the sets. In the first set, moving right gesture is performed with the index finger and the middle finger whereas in the second set, moving to the right gesture is performed with the help of two index fingers. For moving to the left, the gesture for this in the first set is performed using index finger, middle finger and the ring finger whereas in the second set, this gesture is performed by using two thumbs. For the AGV, the first set of predefined gestures were right, left, forward and backwards and the second set of gestures also contained the same type of gestures. The difference in both the sets of the gestures is the same as that of the robot arm i.e., they only differ in the left and right gestures in each list. For the Programming Interface, both set of gestures contained the gestures of selecting, go back and run code. The difference was only in the go back gesture. In the first set, the go back gesture was performed with two hands and in the second set, it was performed with two thumb fingers. For the robot arm mode, it is observed that the users took less time to perform the first set of gestures than the second set of gestures. This could be because the random gestures generated might have been easier to perform in the first set of generated gestures than the second set. Similarly, it was observed that the first set of gestures generated for the AGV and the Programming Interface were easier to perform than the second set of gestures.

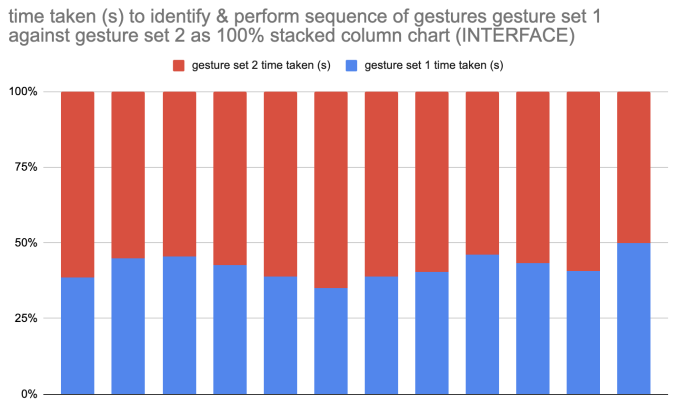


Figure 9: Programming interface accuracy test

3.4 Big-5 Inventory for Emotional Response

The Big Five Inventory (BFI) is a self-report measure of the five broad dimensions of personality, which are often referred to as the "Big Five" or the "Five Factor Model" of personality. The five dimensions are:

- **Openness to experience**: characterized by traits such as imagination, creativity, and appreciation for art and beauty.

- **Conscientiousness**: characterized by traits such as organization, dependability, and self-discipline.
- **Extroversion**: characterized by traits such as sociability, assertiveness, and enthusiasm.
- **Agreeableness**: characterized by traits such as trust, altruism, and cooperation.
- **Neuroticism**: characterized by traits such as anxiety, emotional instability, and moodiness.

The BFI is a widely used measure of personality in both research and applied settings[8]. It consists of 44 items that are rated on a 5-point Likert scale, with responses ranging from "Strongly disagree" to "Strongly agree." The BFI is a brief, easy to use and quick instrument, it's reliability and validity have been demonstrated in multiple studies across different cultures and populations. It is used in research, in the workplace to evaluate job candidates, and in personal development and therapy. The BFI questions were given to the 12 users to measure their personality and understand why they made certain decisions or have strong opinions on some questions. An analysis was made and it was found that 50% of the users have a conscientious along with an agreeable personality which shows that majority of our test users were willing to try new things and had quite some capability to think outside the box. In addition to that, the users were also helpful during the whole process. Further, only 8.3% of the users had an extroverted personality.
Out of the results, we were able to categorize people to being agreeable and not, which correlated with the acceptance rates of the new gestures. 17% of the people that are more agreeable choose to favor the old gestures compared to the new ones, and the people that are not as agreeable did not favor old nor the new gestures.

3.5 Boredom and Frustration test

All interviewees were asked about their emotions before and after doing the list of gestures for the Robot Arm, AGV and the Programming Interface. They had to classify the feelings of the Flow Theory (Frustration and Boredom) with a grade between 1 and 10. The same questions were answered before and after doing the list of gestures for the following reason: The feelings of the person may be different before and after the experiment.

In order to correctly study the influence of the gestures on emotions, for every emotion and every set of gestures, the rating given by each user after testing the gestures is subtracted to that before testing the gestures. Negative values represent an increase in that feeling by the gesture and vice-versa. Then the data is linearly translated for graphical purposes.

In the case of the robot arm, the following figure is the confusion matrix for the Theory of Flow:

	FRUSTRATION	BOREDOM
Predefined Gestures	5.25	5.3333
New Gestures	3	6.5

Table 1: Confusion matrix for the robot arm

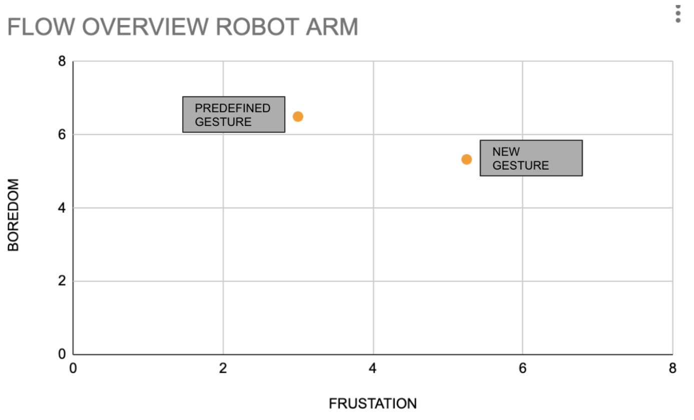


Figure 10: Confusion matrix plot for the robot arm

It can be observed a better engagement occurs in the predefined gestures.

	FRUSTRATION	BOREDOM
Predefined Gestures	2.166667	2.3333
New Gestures	3.83	3.83

Table 2: Confusion matrix for the AVG

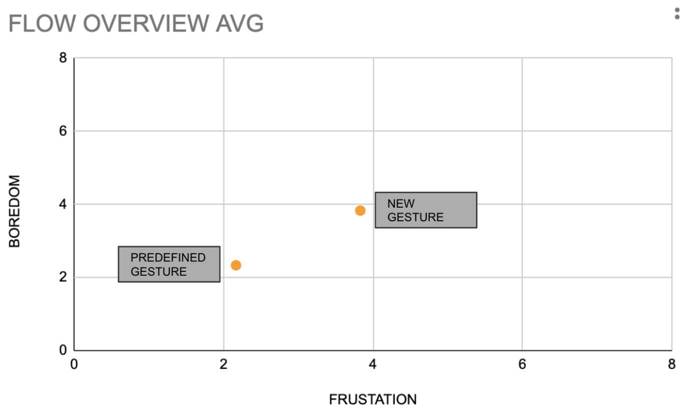


Figure 11: Confusion matrix plot for the AVG

It can be observed a better engagement occurs in the predefined gestures.

	FRUSTRATION	BOREDOM
Predefined Gestures	3.33333	2.916667
New Gestures	6.16667	3.83

Table 3: Confusion matrix for the Programming interface

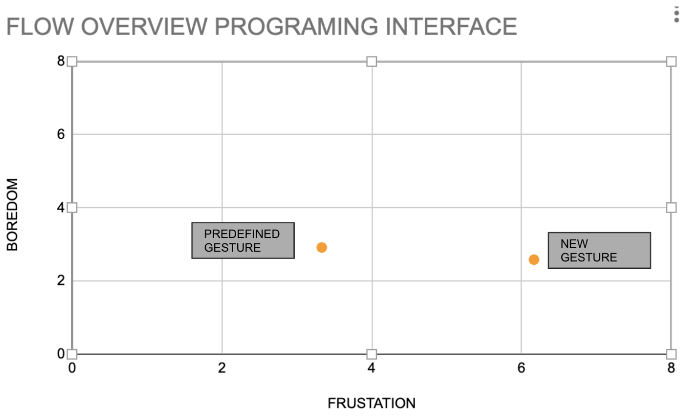


Figure 12: Confusion matrix plot for the Programming

It can be observed a better engagement occurs in the predefined gestures.

3.5.1 Mann-Whitney tests

The Mann-Whitney test is a non-parametric statistical test used to compare two independent groups of observations. It is an alternative to the t-test when the assumptions of normality and equal variances are not met. The test is based on calculating the rank sum for each group and then comparing the two sums to determine if they come from the same population. [9]

In the Mann-Whitney test, no assumptions are made about the distribution of the data, making it more robust against violations of the assumptions of the t-test. However, it has less statistical power than the t-test when the assumptions are valid.

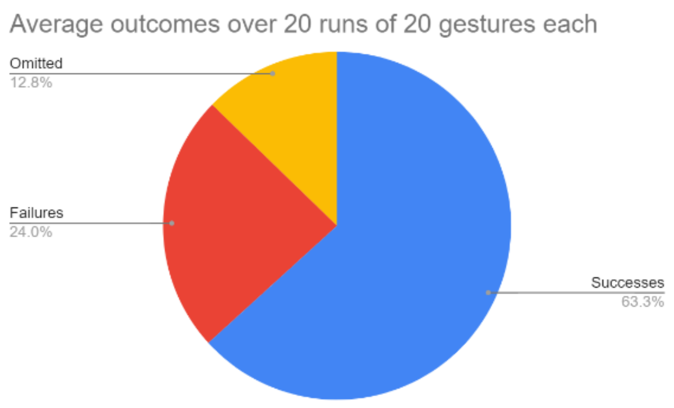
Six Mann-Whitney tests were applied, for every mode and emotion. The alpha that was used is 0.1 and the critical value 42. The results of the tests are shown in the following table:

	alpha	U1	U2	MIN	C V
F RA	0.1	59.5	84.5	59.5	42
F A	0.1	21	123	21	42
F PI	0.1	141	3	3	42
B RA	0.1	104	37.5	37.5	42
B A	0.1	37.5	106.5	37.5	42
B PI	0.1	67	77	67	42

Table 4: Result for the Mann-Whitney test

- **Robot arm**: The Mann Whitney test shows that users feel more bored on the second list of gestures. However, it can not be concluded that people get more frustrated in the first list of gestures, as the critical value is lower than MIN (U1, U2).
- **AVG**: People get more frustrated in the first list of gestures and people get more bored too in the first list of gestures
- **Programming interface**: The Mann Whitney test shows that users feel more frustrated in the second list of gestures and we cannot conclude that people get more bored in the first list of gestures because the critical value is lower than MIN (U1,U2).

4 Discussion



For the initial quantitative experiments involving testing the accuracy of correct identification of gestures, the implementation proved to be sufficiently accurate. During the initial testing of the accuracy of gesture detection it was found that the majority (63.3%) of the gestures were recognized successfully, 24% of the gestures were incorrectly identified and 12.8% of gestures were omitted (Figure 4)).

During our quantitative testing qualitative observations were made. It was observed that the angle of the camera plays an important role in the correct identification of the gestures as should ideally be perpendicular to the person. The lighting also played an important role as a light behind the user causes lens flare which impedes the correct identification of gestures. Finally, a solid background, especially of a dark one aids in the correct identification of gestures because there is a higher contrast between the hand being detected. We also found that the dual thumbs up required an orientation of the knuckles in which the second knuckles are hidden and the first knuckles are visible. Additionally, for the fist gesture it was critically important that the front fingers were visible to the camera to identify the gesture correctly.

5 Conclusion

Looking back at the problem statement discussed in the introduction, this paper focused on a possible implementation of custom gestures that can be used for young inexperienced programmers to easily interact with an insect farm and control the parts that constitute it. On the implementation side, the tools introduced in this paper are CVZone, MediaPipe and OpenCV. In the future, this system may be improved with further research of other frameworks. On the other hand, the physical setup will consist of four cameras connected to a projector, that enable users to view both themselves on the screen, as well as the actual insect farm setup. The results of the experiments point towards the fact that predefined gestures can be performed by users, with high accuracy, without a technical background. It also shows that users think of the gestures as being very interactive as they are not easily bored while performing new gestures. In conclusion, the innovative features that this paper discusses are hand tracking implementation including 8 custom gestures for 3 different modes. This paper also explores real-time dual feedback system sounds and visuals as well as having a flask webhooks for sending and receiving queries to/from other developers involved in the control of the insect farm.

References

1. *Corosect* Dec. 2022.

2. Moryossef, A. *et al. Evaluating the Immediate Applicability of Pose Estimation for Sign Language Recognition* 2021.

3. *CV zone documentation* Jan. 2023.

4. *Feature detection*

5. *Mediapipe documentation*

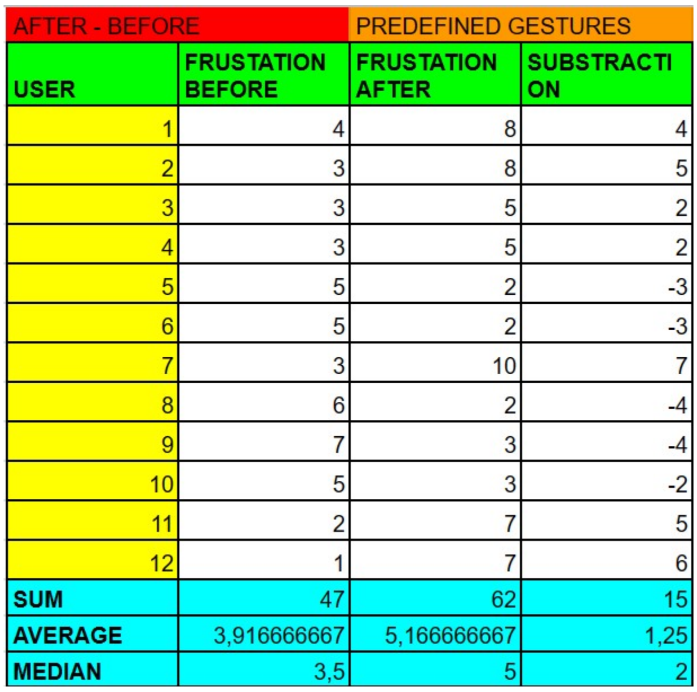
6. *Blockly | google developers*

7. Boonstra, L. in *The Definitive Guide to Conversational AI with Dialogflow and Google Cloud* 203–267 (Springer, 2021).

8. *Handbook of personality : theory and research.* Fourth edition / edited by Oliver P. John, Richard W. Robins. eng (The Guilford Press, New York, 2022 - 2021).

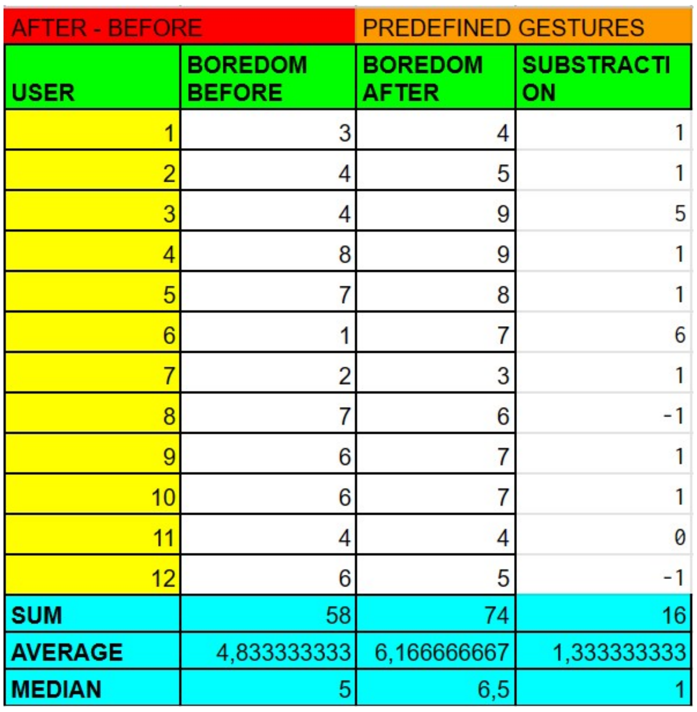
9. McKnight, P. E. & Najab, J. Mann-Whitney U Test. *The Corsini encyclopedia of psychology,* 1–1 (2010).

Appendix



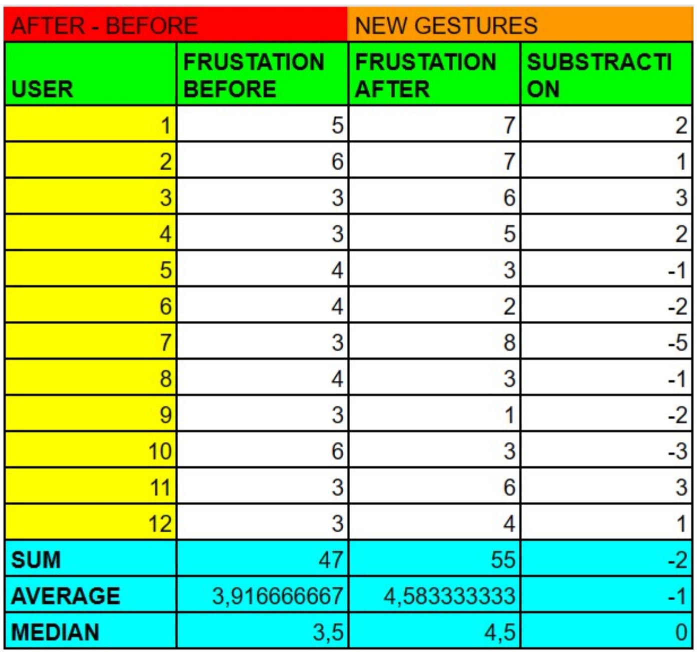
AFTER - BEFORE		PREDEFINED GESTURES	
USER	FRUSTATION BEFORE	FRUSTATION AFTER	SUBSTRACTI ON
1	4	8	4
2	3	8	5
3	3	5	2
4	3	5	2
5	5	2	-3
6	5	2	-3
7	3	10	7
8	6	2	-4
9	7	3	-4
10	5	3	-2
11	2	7	5
12	1	7	6
SUM	47	62	15
AVERAGE	3,916666667	5,166666667	1,25
MEDIAN	3,5	5	2

Figure 13: Robot Arm - Frustration before and after for first set of gestures



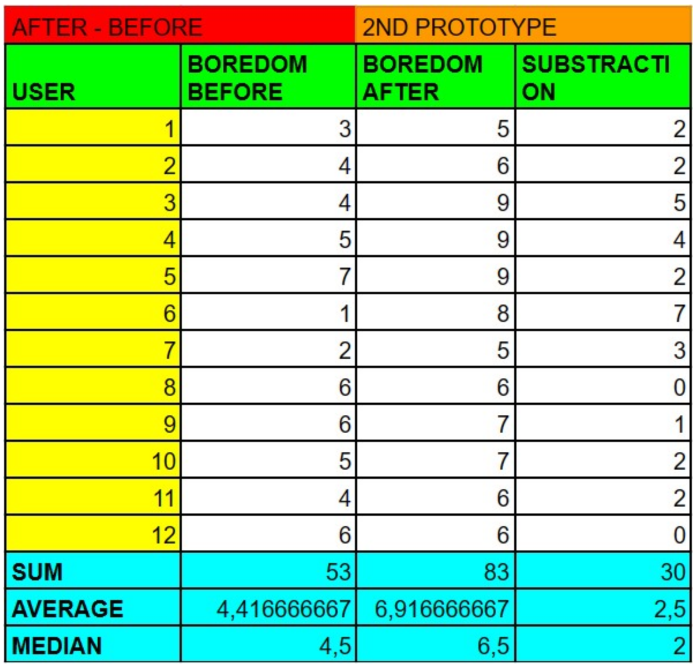
AFTER - BEFORE		PREDEFINED GESTURES	
USER	BOREDOM BEFORE	BOREDOM AFTER	SUBSTRACTI ON
1	3	4	1
2	4	5	1
3	4	9	5
4	8	9	1
5	7	8	1
6	1	7	6
7	2	3	1
8	7	6	-1
9	6	7	1
10	6	7	1
11	4	4	0
12	6	5	-1
SUM	58	74	16
AVERAGE	4,833333333	6,166666667	1,333333333
MEDIAN	5	6,5	1

Figure 14: Robot Arm - Boredom before and after for first set of gestures



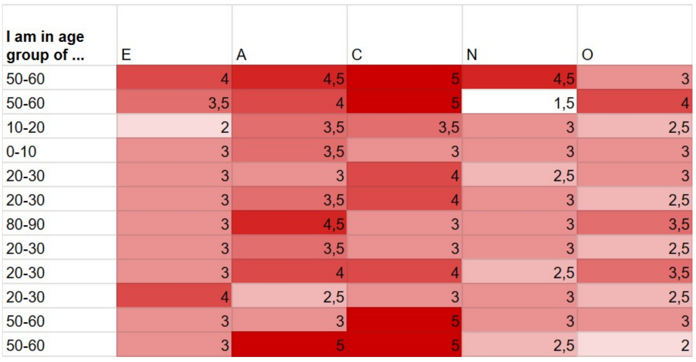
AFTER - BEFORE		NEW GESTURES	
USER	FRUSTATION BEFORE	FRUSTATION AFTER	SUBSTRACTI ON
1	5	7	2
2	6	7	1
3	3	6	3
4	3	5	2
5	4	3	-1
6	4	2	-2
7	3	8	-5
8	4	3	-1
9	3	1	-2
10	6	3	-3
11	3	6	3
12	3	4	1
SUM	47	55	-2
AVERAGE	3,916666667	4,583333333	-1
MEDIAN	3,5	4,5	0

Figure 15: Robot Arm - Frustration before and after for second set of gestures



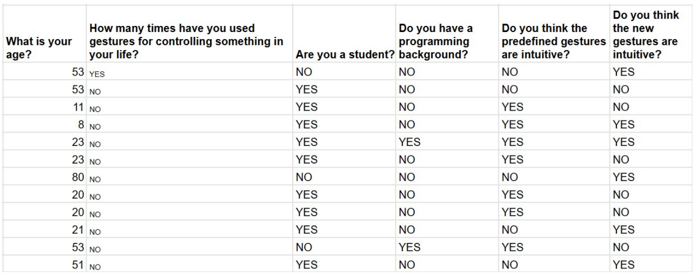
AFTER - BEFORE		2ND PROTOTYPE	
USER	BOREDOM BEFORE	BOREDOM AFTER	SUBSTRACTI ON
1	3	5	2
2	4	6	2
3	4	9	5
4	5	9	4
5	7	9	2
6	1	8	7
7	2	5	3
8	6	6	0
9	6	7	1
10	5	7	2
11	4	6	2
12	6	6	0
SUM	53	83	30
AVERAGE	4,416666667	6,916666667	2,5
MEDIAN	4,5	6,5	2

Figure 16: Robot Arm - Boredom before and after for second set of gestures



I am in age group of ...	E	A	C	N	O
50-60	4	4,5	5	4,5	3
50-60	3,5	4	5	1,5	4
10-20	2	3,5	3,5	3	2,5
0-10	3	3,5	3	3	3
20-30	3	3	4	2,5	3
20-30	3	3,5	4	3	2,5
80-90	3	4,5	3	3	3,5
20-30	3	3,5	3	3	2,5
20-30	3	4	4	2,5	3,5
20-30	4	2,5	3	3	2,5
50-60	3	3	5	3	3
50-60	3	5	5	2,5	2

Figure 17: Big 5 Inventory



What is your age?	How many times have you used gestures for controlling something in your life?	Are you a student?	Do you have a programming background?	Do you think the predefined gestures are intuitive?	Do you think the new gestures are intuitive?
53	YES	NO	NO	NO	YES
53	NO	YES	NO	NO	NO
11	NO	YES	NO	YES	NO
8	NO	YES	NO	YES	YES
23	NO	YES	YES	YES	YES
23	NO	YES	NO	YES	NO
80	NO	NO	NO	NO	YES
20	NO	YES	NO	YES	NO
20	NO	YES	NO	YES	NO
21	NO	YES	NO	NO	YES
53	NO	NO	YES	YES	NO
51	NO	YES	NO	NO	YES

Figure 18: Questionnaire about the users