ColEmo: A Flexible Open Source Software Interface for Collecting Emotion Data

Mohammad Hasan Rahmani
IDLab-Faculty of Applied Engineering
University of Antwerp-imec
Antwerp, Belgium
mohammad.rahmani@uantwerpen.be

Rafael Berkvens
IDLab-Faculty of Applied Engineering
University of Antwerp-imec
Antwerp, Belgium
rafael.berkvens@uantwerpen.be

Maarten Weyn
IDLab-Faculty of Applied Engineering
University of Antwerp-imec
Antwerp, Belgium
maarten.weyn@uantwerpen.be

Abstract—**Data collection is a critical challenge in Emotion Recognition (ER), especially as demand grows for in-the-wild data that includes contextual information. The collection of data is continuously needed to investigate new ER modalities, sensors, stimuli, models, and methods. This necessitates the development of tools and frameworks that facilitate emotion data collection. In this paper, we introduce ColEmo, an open-source software interface for collecting emotion data. ColEmo was developed using Flutter allowing it to be compiled for both desktop and mobile devices. The architecture and interface of ColEmo provides a high degree of flexibility to be customized or extended to suit specific experiment requirements. We tested ColEmo in an ER data collection study which was extended to include Voice Activity Detection (VAD) and motion context, demonstrating its effectiveness in lab environments. Further investigation is recommended to evaluate ColEmo's potential for in-the-wild data collection setups.**

Index Terms—**Data Collection, Affective Computing, Emotion Recognition, Context Awareness, Open Source Software**

I. Introduction

Emotions are among the most influential factors in people's lives, which can affect many of their activities, communications, and daily routines [1]. They are derived from a variety of external and internal human body data. External representations of emotion include a range of human perceptible emotion expressions. Facial [2], [3] and vocal [4], [5] modalities are among the most prominent studies in this category, which have received much attention from the research community. However, external emotion representations are not limited to facial and speech data. Tactile expressions [6], movements [7], [8], and textual demonstration [9] are also among the external emotion representations. Internal emotion representations, on the other hand, consist of a set of physiological signals. Electroencephalogram (EEG), Elegtromyogram (EMG), Heart Rate (HR), Heart Rate Variability (HRV), Respiration Rate (RR), and Electro-dermal Activity (EDA) are among the commonly used internal modalities for emotion recognition [10].

Emotion Recognition (ER) is constantly being improved through the development of new methods based on Artificial Intelligence (AI), as well as the creation of new datasets. Preparing a dataset for ER requires a number of decisions regarding several requirements. These requirements include (but are not limited to): modalities involved, sensors, experiment environment, stimuli, emotion model, ground truth verification, number and conditions put on subjects. Table I lists these requirements with some of their possible choices, taken from the current literature.

TABLE I
Requirements for collecting datasets in Emotion Recognition (ER). These include the selection of appropriate modalities and corresponding sensors, stimuli that elicit intended affective states, an emotion model and rating scale for human annotations, experimental setup, participant recruitment criteria, and respecting research ethics in all stages.

Requirement	**Options**
Modalities	Facial; Speech; Gait; Gesture; Brain signals; Heart activity; EDA; EMG; EOG; RSP
Sensors	Medical; Commercial
	Local memory; Connected (wired, wireless)
Stimuli	Picture; Music; Video; Environment; Memories
Emotion model	Discrete emotions; Dimensional emotions
Rating scales	PANAS [11]; SAM [12]; DES [13]
Experiment setup	Lab (controlled); In the wild
Conditions on participants	Quantity; Age groups; Gender balance; Culture
Research ethics	Depending on experiment content and design

Acronyms used in the table: DES: Differential Emotion Scale, EDA: Electro-dermal Activity, EMG: Elegtromyogram, EOG: Electrooculogram, PANAS: Positive and Negative Affective Scheme, RSP: Respiration, SAM: Self-Assessment Manikins,

The multiple possible combinations of the individual parameters reduce the chance of finding a perfect match out of the available emotion datasets. Therefore, for many future ER studies, proprietary data will have to be collected to keep updated with the advent of new ground truth verification methods, field data collection techniques, and new modalities or sensors. Moreover, as AI is fed with data, in order for ER to reach a higher level of maturity, long way is still to go regarding emotion data collection. All of this necessitates the development of tools and frameworks that facilitate the collection of emotion data.

Collecting data for ER can be a challenging task. Framing the requirements listed in Table I provides some insight into the components that are common to any task of collecting ER data which are in line with the previous datasets on ER [14]–[17]. These common components which are also

This research was funded by UAntwerp BOF-GOA-NudJIT.

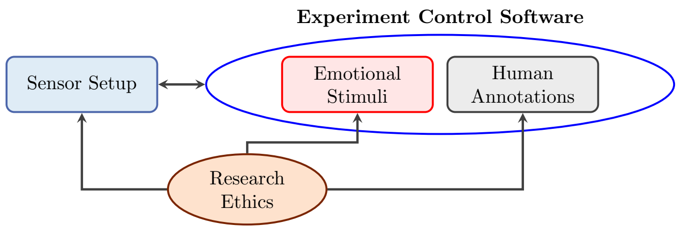

Fig. 1. Common components of any software-based Emotion Recognition (ER) data collection task derived from ER data collection requirements (Table I) and the existing ER datasets [14]–[17]. The software for experiment control is in charge of presenting emotional stimuli, collecting human annotations and communicating with the sensors. Research ethics must be respected in all aspects of the data collection.

briefly demonstrated in Figure 1 include:

1) *Research Ethics*: Respecting research ethics in all detail of the data collection before, during, and after the experiments is vital,
2) *Sensor Setup*: Preparation and setup of the sensors used for data collection is a common part of the experiments which often involves helping participants wear wearable sensors among other setups,
3) *Emotional Stimuli*: Presenting effective emotional stimuli is necessary for eliciting different emotional states in the subjects,
4) *Human Annotations*: Collecting self-reported or other-reported annotations that help getting close to ground truth information for the subject's affective state is crucial.

In addition, *experiment control software*, usually in the form of a User Interface (UI) is required to coordinate the experiments by presenting the stimuli, interacting with user inputs and sensor data throughout the trials. Experiment builder toolboxes help researchers create their own experiment setups and present them to subjects in the form of a UI application [18]–[22]. However, these toolboxes come with limitations such as being platform-restricted, closed source, or requiring paid licenses. We will further discuss these existing software in Section IV (Table II).

In this paper, we present an open-source UI application software developed initially for the ER data collection research. We tested our software within a context data collection study covering emotion, voice, and activity, which we present in this paper. Please note that the main focus of the current paper is to introduce our open-source UI application software, while our dataset itself, along with its detailed data description will form a separate future contribution. We named our software *ColEmo* (abbreviating **Col**lecting **Emo**tions). ColEmo is developed using a cross-platform Software Development Kit (SDK). This helps ER researchers interested in collecting in-the-wild data to compile their app with minimal programming effort for mobile platforms (Android, iOS, or web) according to their own needs. Furthermore, ColEmo is a modular application that can be integrated with different sensors, different stimuli, and/or different human annotation rating scales. Flexibility of ColEmo is threefold: modularity, extensibility via Python Application Programming Interface (API), and cross-platform compatibility. Such versatility makes ColEmo suitable for data collection in research areas that require human data, including sensed data, self-reported data, or third-person annotations. This opens up usefulness of ColEmo in various application domains, such as emotion recognition, context-awareness, psycho-physiological analysis, and more. ColEmo is accessible at: https://gitlab.ilabt.imec.be/emowear/colemo.

The remainder of this paper is organized as follows: Section II first provides an introduction to the features of ColEmo, followed by its underlying architecture. Then a description of its interface within a working framework is given. Section III describes our use case of data collection using ColEmo as a practical example. Section IV discusses the technical aspects of the various elements used and the results of our experience with ColEmo, and provides suggestions for future work. Finally, conclusions are drawn in Section V.

II. ColEmo Application Software

We developed ColEmo using Flutter[1]. Flutter is an open-source SDK for cross-platform UI application design, created by Google. ColEmo is written in Dart[2] programming language. The same codebase can be compiled natively for Android, iOS, Linux, macOS, Windows, and the web platforms. To maintain the modularity of the whole framework, ColEmo can work in conjunction with other pieces of software that handle independent tasks through APIs. ColEmo performs the following operations during the user data collection session:

- Provides instructions that guide the user through the programmatically-defined phases of the data collection session,
- Presents emotional stimuli to elicit different emotional states in the participants,
- Collects user input data i.e. the pre-study questionnaire, intermittent self-assessment surveys, and microphone recordings,
- Generates timestamped Message Queuing Telemetry Transport (MQTT) log messages for every application-related event.

In this section, we make a closer look into the capabilities of ColEmo as our proposed user data collection interface for ER applications.

A. Architecture

Flutter apps are made from widgets as their central class hierarchy of the framework. Although widgets are immutable by definition, mutable states can be associated with them resulting in stateful widgets. Within the architecture of ColEmo, BLoC state management library[3] is used to control the major parts of the UI state.

[1]https://flutter.dev/
[2]https://dart.dev/
[3]https://bloclibrary.dev/

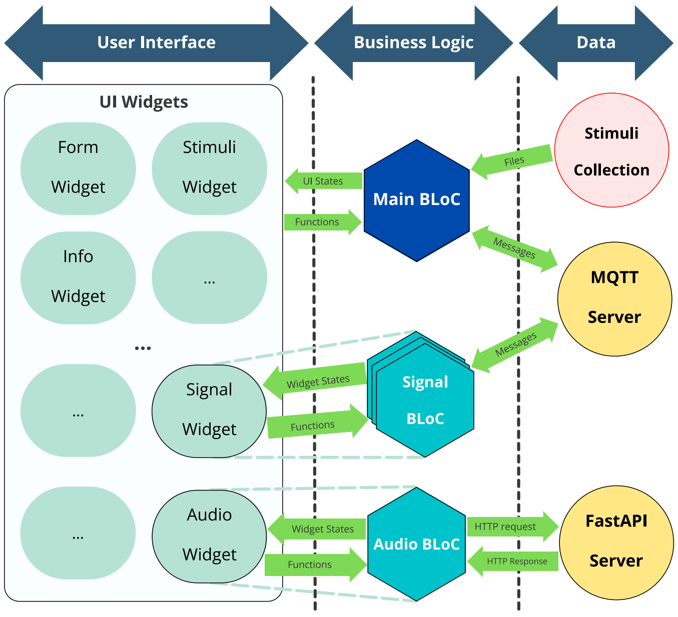

Fig. 2. The underlying architecture of ColEmo. Business Logic Component (BLoC) state management library is used to separate UI from the business logic. The "Main BLoC" controls the flow of the experiments. The "Signal BLoCs" take care of MQTT messages containing sensors data. The "Audio BLoC" controls the state of a special widget that is only responsible for the voice data acquisition phase.

Figure 2 illustrates the underlying architecture of ColEmo. Within the implemented architecture, ColEmo can be explained in three top-view segments: the user interface, the business logic, and the data handling. Ideally, in the context of the BLoC pattern, the user interface segment should consist only of elements that manipulate UI and have nothing to do with the app's business logic. Therefore, the app's business logic can be separated from the UI design and manipulated elsewhere. The same strategy is applied in ColEmo design.

Three major BLoC types control ColEmo's states. The Main BLoC, the Signal BLoCs and the Audio BLoC. We will take a look at each of them in the following paragraphs.

The Main BLoC plays a key role in the app. The order of the UI widgets that appear on the screen, and thus the flow of the experiments, is programmed in the Main BLoC. The Main BLoC controls which widget is presented and switches it based on either the widget's function calls or the BLoC's own internal elapsed time. It also decides which stimulus to present to the user.

The Signal BLoCs are in charge of communication with the sensors through the MQTT server. The idea is that each Signal BLoC would be in charge of managing messages from one sensor. By subscribing to the topics where different sensor data are published, Signal BLoCs help providing a visual feedback on whether or not the sensor data are continuously present during the experiments.

The Audio BLoC is instantiated with the creation of the audio widget in the app's widget tree. Within the audio widget, the subject can record their voice, listen to the recording, manually review, edit, or remove individual chunks of the processed recording, repeat any of the previous steps, or submit the reviewed timestamps. Since the audio widget is relatively more complex than the other widgets, a separate BLoC manages the states of this widget.

B. Interface

Emotion data is always collected in a complete framework consisting of a package of the necessary hardware and software. We also used ColEmo inside a data collection framework (Figure 3) that is further explained in Section III. The data interface provided by ColEmo application is crucial in understanding its role within the framework. Figure 3 illustrates the flow of various data within our data collection framework, showing the setup of an MQTT server and a FastAPI server to communicate with the ColEmo application. Additionally, the experimenter can remotely monitor the experiments' progress and intervene using control commands if necessary. Dashed lines in the figure indicate parts that have not been implemented or used in the final used framework.

A Voice Activity Detection (VAD) system is developed as a separate Python API using the FastAPI toolkit[4]. To host this API, a FastAPI server is run locally with which the Audio BLoC communicates through HTTP requests. The API is programmed to process retrieved audio recordings and respond with the timestamps indicating the beginning and the end of the voiced chunks of the recording. Once the processing results arrive, the Audio BLoC of ColEmo updates the UI with the detected audio chunks, which can then be manually reviewed and accepted, edited, or removed. If the chunks are verified by the user, corresponding timestamps are published to their dedicated MQTT topic.

MQTT has been selected as the main message handling system in our proposed framework. ColEmo needs to connect to an MQTT server to log its data. Timestamped logs of all events, user input data including the self-assessment ratings and the pre-study questionnaire, as well as the verified voice annotations from the VAD system are published to their corresponding MQTT topics. Consumers of these topics include the Logger application and a monitoring application on the experimenter's laptop (over the network). Within MQTT, nodes can subscribe to their topics of interest. The Logger application subscribes to all topics (using the '#' wildcard) and stores all messages in an SQLite database file. Therefore, the database file is the final product of all the data that ColEmo collects.

The framework should also collect data of the intended modalities from the subjects, i.e. the data of the sensors. In our final setup, all sensors are set to store data in their internal memory. However, the dashed lines in Figure 3 show the path that sensor data could take to be stored and/or visualized if an online sensor data collection scheme were in place. This path has been completely implemented and tested confirming its functionality; however, it was eventually set aside in our

[4]https://fastapi.tiangolo.com/

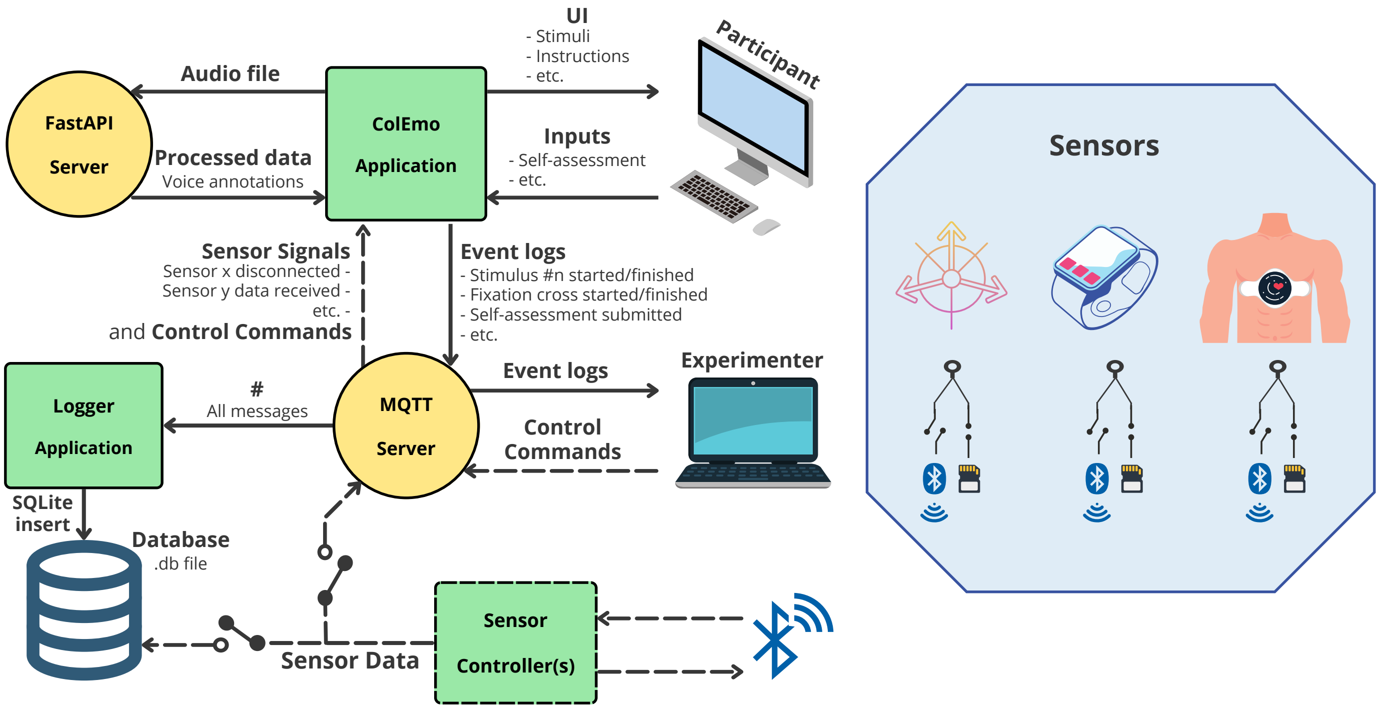

Fig. 3. Illustration of the data flow within our data collection use case. The green rectangles are standalone application software. The FastAPI server is an optional extension used to provide Python API for audio processing. Sensor data can be stored in the sensors' internal memory or streamed via Bluetooth. The sensor controller(s) can be both desktop or smartphone apps. The data paths with dashed lines were ignored in the final presented use case.

use case to reduce the complexity of the overall system among other reasons (Sections III-A and IV).

III. Data Collection Use Case

In this section, we describe our data collection as a use case for ColEmo. Providing an actual use case helps future users to gain familiarity with the potential capabilities of ColEmo through a practical example and to learn about the existing elements of the first released version. The modularity of the framework allows researchers to integrate their own hardware and experiment protocol with ColEmo, as long as they adhere to its interface. The purpose of this section is to show the usefulness of ColEmo in a practical data collection example. Our data collection use case is extended to include extra contextual modalities other than emotions: voice activity as a hint to person's social context, and gait as an indication of person's activities. As stated in Section I, the collected data itself will be a future contribution which is still in progress.

A. Measurement Hardware

As for the ultimate purpose of context-awareness, we only used mobile wearable sensors in our experiments. In a previous survey paper, we showed the usefulness of chest-worn Inertial Measurement Units (IMUs) for obtaining data in several areas, including ER [23]. To further explore this, we used two *ST SensorTile.box* sensors, one on the sternum and the other on the back of the subjects, exactly on the opposite side of the first sensor. Both boxes are pressed against the body via a manually designed elastic strap around the chest at sternum height. Also, we used two different wearable sensor sets to collect physiological signals: *Zephyr Bioharness 3* and *Empatica E4*. Zephyr Bioharness 3 is equipped with Electrocardiography (ECG), Respiration (RSP), and inertial sensors, embedded on an elastic strap [24] that must be worn on the chest and pressed against skin. Empatica E4 is a wristband capable of sensing Photoplethysmogram (PPG), EDA, Temperature (TMP), as well as inertial data. This device is frequently used in the literature for emotion and stress detection [25]–[28]. Figure 4 shows how the mentioned wearable devices were worn in this study.

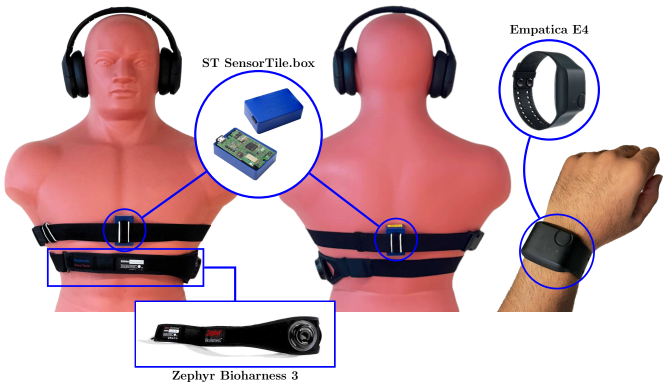

Fig. 4. Utilized wearable hardware, including noise-cancelling headphones, chest strap holding two ST SensorTile.boxes, Zephyr Bioharness 3 strap, and Empatica E4 wristband.

Although all of the sensors used offer wireless data streaming via Bluetooth, we set them to record sensory data on their internal memory that can later be read out via a computer device. This is to minimize the risk of data loss due to possible signal interference or poor connection and to maximize the performance of the sensors, especially considering their battery capacity.

B. Experiment Protocol

The experiments lasted about two hours and were performed in a room at our institute. Forty nine subjects participated in the experiments. The subjects were first informed about all details of the data collection process. They were asked to fill out and sign an informed consent if they agreed to participate in the experiments. The ColEmo app was run on a laptop dedicated to the experiments, which we refer to as the participant laptop. To avoid unnecessary complexity, the FastAPI server, the MQTT server, and the Logger application were all run on the participant laptop. The participant laptop was connected to an external 24-inch monitor running ColEmo in full-screen mode. Next, the subject was asked to sit on a chair in front of the monitor. First, they filled out a pre-study questionnaire implemented inside the UI. Next, they were helped to put on the sensors and perform three consecutive jumps to facilitate sensor synchronization. They sit back in front of the monitor afterwards. The subjects were then asked to read ten sentences that appeared on the screen. At this stage, we temporarily recorded their voice to perform a VAD task as an extension to the ER purpose. Their voice served to mark precisely the beginning and end of their vocal activity. A completely separate Python API was used to process the recording. The time markers were then immediately displayed on ColEmo for a manual check by the experiment supervisor. The provision of the VAD extension is useful to demonstrate ColEmo's ability to integrate with additional modules implemented with different codebases.

The main experiments were composed of 38 repeated sequences with different stimuli. In each sequence, a one-minute stimulus video was shown on the screen. Next, the well known Self-Assessment Manikins (SAM) were used for the subject's self-assessment. Finally, ColEmo asked the subject to stand up, walk a few steps in a certain route, and return to their seat for the next sequence. Walking was added for various reasons, above all, to make the experiments more similar to real-world scenarios where physical activity can significantly influence physiological signals [29], [30]. Before the main experiments began, subjects were shown a trial sequence to familiarize them with the procedure they will face. In addition, a five-minute break was provided after the 19th sequence, during which subjects were offered refreshments that did not contain caffeine or alcohol.

IV. Results and Discussion

ColEmo provides a responsive, beautiful, clear, and easy-to-use UI for ER data collection. It covers all the necessary components for collecting ER datasets while being open-source, adaptable, and extensible. Figure 5 shows a few screenshots from inside ColEmo. Earlier in Section I, we presented four components common to any ER data collection task (Figure 1). Now, the screenshots are selected to demonstrate ColEmo's relation to each of those components i.e. communicating with sensor devices (Figure 5a), providing stimulus content (Figure 5b), collecting subjective self-reported assessments (Figure 5c), and obtaining consents in alignment with research ethics (Figure 5d). It is important to mention that obtaining participant consent for collecting their data does not address all necessary ethical concerns, but it is a helpful step.

Table II lists the existing software used for collecting ER data. In the second column it indicates whether each software provides a tool for building the experiment structure that participants will follow. Although ColEmo does not offer the convenience of easily building experiment structures without modifying its codebase, it provides other unique benefits that have not been available in a single application software before. For example, ColEmo can be compiled for multiple platforms from the same codebase, which enables in-the-wild data collection studies. Its support for the MQTT communication interface allows for the integration of sensors directly or via standalone third-party apps that wrap those sensor data with MQTT interface. Additionally, its capability of being extended through a FastAPI Python backend provides the opportunity to leverage existing tools that can handle instant processing of requests and response with the results, a capability not offered by the other listed software, which distinguishes ColEmo from them. In our presented use case, we used FastAPI to provide accurate voiced time markers for the extended VAD task.

The ColEmo application was put into practice within the framework presented in section III. In a nutshell and from a technical point of view, ColEmo is a Flutter application, with the BLoC library as its major state management pattern, MQTT as its main communication gateway, and FastAPI as its extended backend support (Figure 3). Let's discuss Flutter, BLoC, MQTT, and FastAPI.

Flutter enables cross-platform UI application development. Consequently, ColEmo's codebase can be compiled natively for any target platform with high performance. Since ColEmo is an open-source application, made to adapt with various requirements in ER data collection, platform independence is a big plus, especially with ER moving more and more towards in-the-wild research scenarios. The ability to collect data from computers, tablets, and smartphones better supports the future needs of ER. However, there is currently no single cross-platform SDK that also covers compilation for wearable technology such as smartwatches. This marks a limitation of ColEmo, as it cannot be compiled for wearable technology. However, to overcome this limitation in the short term, MQTT offers the possibility of integration with independent software for wearable technology.

The use of BLoC within the ColEmo architecture separates the UI from the app's business logic. Table I listed part of the wide options that exist for different requirements of an ER data collection task. BLoC ensures that adapting the app to

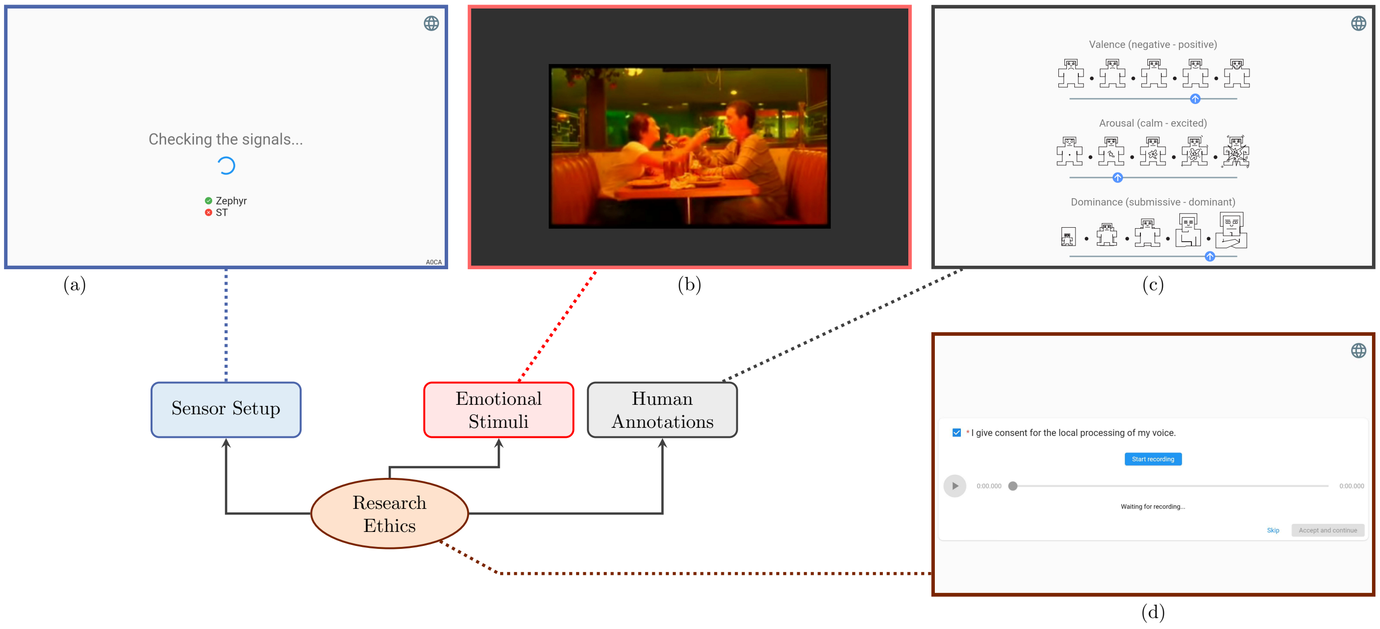

Fig. 5. Sample screenshots from ColEmo app that reflect correspondence to each of the main components of any Emotion Recognition (ER) data collection task previously listed in Figure 1.

TABLE II
COMPARING EXISTING SOFTWARE THAT CAN BE USED FOR COLLECTING EMOTION DATA. ALL THESE SOFTWARE ALLOW FOR PROVISION OF EMOTIONAL STIMULI, COLLECTING HUMAN ANNOTATIONS, AND SYNCHRONIZED SENSOR DATA.

Software	Exp. Builder	Open Source	Platforms						Sensor Integration	API Extensibility	Software Communication Interface
			Windows	Linux	MacOS	Android	iOS	Web			
AWARE [31]		✓				✓	✓		Smartphone sensors	-	MQTT, HTTP
OpenSesame [18]	✓	✓	✓	✓	✓				Limited plug-ins	-	-
Tobii Pro Lab [19]	✓		✓	✓	✓				Limited partners	-	-
Presentation® [20]	✓		✓						Limited products[1]	-	-
[21][2]			✓						Limited products[1]	-	-
E-Prime® [22]	✓		✓						Limited products	-	-
ColEmo (current work)		✓	✓	✓	✓	✓	✓	✓	Through MQTT or offline[3]	Python API support	MQTT, HTTP

1- Products that support communication through Presentation's general port I/O facilities.
2- A Presentation® script.
3- Offline sensors must be equipped with accelerometers.

different choices can be done easily by properly separating the software elements. Removing the existing experiment phases, replacing them with other designs, or adding new phases in between, can all be done within the main BLoC of the app.

MQTT is a common communication protocol for Internet of Things (IoT) [32]. Considering the rising trend for in-the-wild ER research scenarios, MQTT is interesting as it facilitates lightweight communications over the internet. MQTT provides three levels of Quality of Service (QoS) which is a great feature for reliable delivery of messages [33]. Despite the great capabilities of MQTT for sensor control in IoT, it is not designed for collecting data from multiple sensors with high sample rates. To this end, and as a future improvement, we propose the integration of Robot Operating System (ROS) a framework from the robotics community. ROS works similarly to MQTT in a publish/subscribe messaging scheme and is perfectly suited to handle large amounts of sensor data [34].

We had the following two options for collecting sensor data: storing data in internal memory (offline mode) or streaming via Bluetooth (online mode). Each option has its own advantages and disadvantages. In both cases, all sensor data must be synchronized. We will continue covering both modes in the following.

In online mode, sensor data is sent in packets. In this way, the moment a packet is received can be correlated with the moment data was actually sensed. However, different communication speeds can result in incorrect data timing, especially with respect to other sensors. We also noticed some packet loss due to interference from other existing Wi-Fi and Bluetooth signals in the room. Moreover, there were individual challenges with each sensor in online mode, e.g., the SensorTile boxes ran out of battery very quickly and the Bioharness offered less data sampling rates in online mode than in offline scheme.

In offline mode, sensor data is stored in internal memory and read out afterward. Offline mode ensures higher security in terms of data loss but does not provide constant feedback that the data is being recorded correctly. In this mode, each sensor stores its data in its own internal memory without being aware of the other sensors. Therefore, synchronization is still necessary in offline mode which must be done afterward.

To avoid any probable signal interference and packet loss in the online approach, we opted for the offline mode, withdrawing from the advantage of constant feedback. In [25] an Empatica E4 and a chest strap are synchronized manually by two double tap gestures at the beginning and the end of the experiments. As all our sensor units are also equipped with IMUs, we utilized a similar approach, taking advantage of synchronized movement gestures during the experiments. To obtain meaningful timestamps from ColEmo, synchronization between the participant's laptop and the sensors was also required. This was accomplished by updating the Bioharness sensor clock with that of the laptop prior to each data collection. However, since the sensor has its own Real Time Clock (RTC), the clocks can get drifted over time. Although such drift is negligible in our short-duration scope, synchronization remains a challenge for long-term and in-the-wild research scenarios.

Finally, we turn to the potential applications of ColEmo for future research. ColEmo has the basic elements required for ER data collection including the presentation of stimulus content, and the collection of the self-assessments (Section III). Beyond these, we extended our use case towards more context awareness by adding a VAD task and a walking phase to the ER basics. The provision of FastAPI backend support can extend the capabilities of ColEmo to any use case where integration with Python toolboxes is intended (e.g., image or audio processing). In addition, BLoC provides the ability to easily add or remove elements from the UI. ColEmo is open-source and can be downloaded, adapted, and used without restriction, allowing researchers to use it for any type of user study, such as affective, cognitive, or context-aware research.

V. Conclusion

In this paper, we presented an open-source cross-platform application software for emotion data collection studies called ColEmo[5]. Due to its versatility, ColEmo is especially useful for applications of emotion data collection where context information is also needed. ColEmo provides a graphical interface that elicits intended emotional states in subjects and collects their input such as self-assessments. Availing of the well-known BLoC state management, FastAPI backend support, and MQTT communication bus, makes ColEmo easily customizable depending on the experimenters' requirements. Within the proposed framework for ColEmo, modality data can be collected either by independent controller software that provides an MQTT interface or through the internal memories of the sensors. We used ColEmo in a data collection study with forty nine volunteers and explained our experimental setup as an example use case. We are currently processing the collected data and will publish the results in the near future.

[5]https://gitlab.ilabt.imec.be/emowear/colemo

To further improve the software, we propose the integration in ROS, a framework from the robotics community. We also suggest investigation of sensor synchronization methods for long-term data collection scenarios where sensors may go out of sync over time. Currently, we are not planning any experiments in-the-wild ourselves, but we encourage other researchers to use and improve our software in such settings. We look forward to seeing our software used in future experiments and provide support for this.

Ethical Impact Statement

In this paper we presented ColEmo, a software application that facilitates control of the experiments for emotion data collection. Since our software is intended to collect data from human subjects, it will inevitably raise ethical concerns such as privacy, confidentiality, informed consent, and potential harm to participants that must be addressed. Also we will address how we coped with ethical concerns during our presented data collection use case (i.e. Section III).

ColEmo can be compiled for desktop and mobile devices, providing the opportunity for in-the-wild data collection. Depending on the chosen combination of stimulus contents and the coupled sensors, it can cover data collection for a range of applications from affective computing to context awareness and psychological or neurobehavioral sciences. Using this software has the potential risk of having a negative impact such as emotional distress on the participants through misuse of the stimulus contents. The selection of stimulus contents is critical to the intended purpose of the study and its ethical impact. In this regard, sufficiently and accurately informing the subjects about the nature of the presented stimuli or any kind of harm associated with it, is an important task of the experimenters prior to the experiments. An additional workaround can be to arrange order of the presented stimuli in a way that the participant leaves the study with a positive content and thus a positive feeling where possible. This was exactly what we did in our presented use case next to sufficiently informing the participants about the contents in advance. It is worth mentioning that we have excluded our own stimulus contents from the published source code due to copy right issues.

The subjects must be properly informed on the experiment details prior to their participation. Their explicit consent for participation is needed, and their right of withdrawal from the experiments at any stage of the data collection procedure must be informed and respected. Although the experimenter is responsible for these steps, ColEmo provides tools to facilitate participant guidance via "Info Widgets" (see Figure 2) as well as the provision of an initial trial sequence that walks the subject through different phases of the experiment. We encourage researchers to make use of these existing leverages to precisely guide their participants through the study.

Another potential risk associated with data collection using ColEmo is the leak of the data and privacy concerns. This concern is especially heightened in studies that collect personal information, in-the-wild studies, and studies that make use of privacy intrusive sensors like cameras, microphones, or localization services like the Global Positioning System (GPS). To overcome this, ColEmo makes use of the well-established and standardized IoT communication protocol, MQTT. We recommend future researchers to properly secure their MQTT servers and the communication with strong authentication processes. As a future work, encrypting messages before sending them can further increase data security in ColEmo.

The conducted data collection study (Section III) was reviewed in advance by the ethics committees of the University of Antwerp under the file *SHW_22_035* and received a positive decision regarding ethical clearance. The participants were sufficiently informed about the study and their consent was obtained. They were informed of their right to withdraw from the study at any time. It is worth noting that no private data was saved from the participants.

References

[1] P. C. Petrantonakis and L. J. Hadjileontiadis, “Emotion Recognition from Brain Signals Using Hybrid Adaptive Filtering and Higher Order Crossings Analysis,” *IEEE Transactions on Affective Computing*, vol. 1, no. 2, pp. 81–97, 2010.

[2] D. Mehta, M. F. H. Siddiqui, and A. Y. Javaid, “Facial emotion recognition: A survey and real-world user experiences in mixed reality,” *Sensors*, vol. 18, no. 2, p. 416, 2018.

[3] S. M. Moghadam and S. A. Seyyedsalehi, “Nonlinear analysis and synthesis of video images using deep dynamic bottleneck neural networks for face recognition,” *Neural Networks*, vol. 105, pp. 304–315, 2018.

[4] F. Weninger, M. Wöllmer, and B. Schuller, “Emotion recognition in naturalistic speech and language—a survey,” *Emotion Recognition: A Pattern Analysis Approach*, pp. 237–267, 2015.

[5] S. Mirsamadi, E. Barsoum, and C. Zhang, “Automatic speech emotion recognition using recurrent neural networks with local attention,” *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 2227–2231, 6 2017.

[6] A. Schirmer and R. Adolphs, “Emotion Perception from Face, Voice, and Touch: Comparisons and Convergence,” *Trends in Cognitive Sciences*, vol. 21, no. 3, pp. 216–228, 2017.

[7] M. Raja and S. Sigg, “Applicability of RF-based methods for emotion recognition: A survey,” in *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, 2016, pp. 1–6.

[8] H. Zacharatos, C. Gatzoulis, and Y. L. Chrysanthou, “Automatic Emotion Recognition Based on Body Movement Analysis: A Survey,” *IEEE Computer Graphics and Applications*, vol. 34, no. 6, pp. 35–45, 2014.

[9] N. M. Hakak, M. Mohd, M. Kirmani, and M. Mohd, “Emotion analysis: A survey,” in *2017 International Conference on Computer, Communications and Electronics (Comptelix)*, 2017, pp. 397–402.

[10] S. M. Alarcão and M. J. Fonseca, “Emotions Recognition Using EEG Signals: A Survey,” *IEEE Transactions on Affective Computing*, vol. 10, no. 3, pp. 374–393, 2019.

[11] D. Watson, L. A. Clark, and A. Tellegen, “Development and Validation of Brief Measures of Positive and Negative Affect: The PANAS Scales,” *Journal of Personality and Social Psychology*, vol. 54, no. 6, pp. 1063–1070, 1988.

[12] M. M. Bradley and P. J. Lang, “Measuring emotion: the self-assessment manikin and the semantic differential,” *Journal of behavior therapy and experimental psychiatry*, vol. 25, no. 1, pp. 49–59, 1994.

[13] J. J. Gross and R. W. Levenson, “Emotion elicitation using films,” *Cognition & emotion*, vol. 9, no. 1, pp. 87–108, 1 2008.

[14] S. Koelstra, C. Muhl, M. Soleymani, Jong-Seok Lee, A. Yazdani, T. Ebrahimi, T. Pun, A. Nijholt, and I. Patras, “DEAP: A Database for Emotion Analysis ;Using Physiological Signals,” *IEEE Transactions on Affective Computing*, vol. 3, no. 1, pp. 18–31, 1 2012.

[15] M. Soleymani, J. Lichtenauer, T. Pun, and M. Pantic, “A Multimodal Database for Affect Recognition and Implicit Tagging,” *IEEE Transactions on Affective Computing*, vol. 3, no. 1, pp. 42–55, 1 2012.

[16] S. Katsigiannis and N. Ramzan, “DREAMER: A Database for Emotion Recognition Through EEG and ECG Signals From Wireless Low-cost Off-the-Shelf Devices,” *IEEE Journal of Biomedical and Health Informatics*, vol. 22, no. 1, pp. 98–107, 1 2018.

[17] T. Song, W. Zheng, C. Lu, Y. Zong, X. Zhang, and Z. Cui, “MPED: A multi-modal physiological emotion database for discrete emotion recognition,” *IEEE Access*, vol. 7, pp. 12 177–12 191, 2019.

[18] S. Mathôt, D. Schreij, and J. Theeuwes, “OpenSesame: An open-source, graphical experiment builder for the social sciences,” *Behavior Research Methods*, vol. 44, no. 2, pp. 314–324, 6 2012.

[19] “Tobii Pro Lab software,” Tobii Pro AB, Danderyd, Sweden, https://www.tobiipro.com/.

[20] “Presentation® software,” Neurobehavioral Systems, Inc., Berkeley, CA, www.neurobs.com.

[21] C. T. Noto, S. Mahzar, J. Gnadt, and J. S. Kanwal, “A flexible user-interface for audiovisual presentation and interactive control in neurobehavioral experiments,” *F1000Research*, vol. 2, 6 2013.

[22] “E-Prime® software,” Psychology Software Tools, Inc., Pittsburgh, PA, https://pstnet.com/products/e-prime/.

[23] M. H. Rahmani, R. Berkvens, and M. Weyn, “Chest-Worn Inertial Sensors: A Survey of Applications and Methods,” *Sensors*, vol. 21, no. 8, p. 2875, 4 2021.

[24] J. A. Johnstone, P. A. Ford, G. Hughes, T. Watson, A. C. Mitchell, and A. T. Garrett, “Field Based Reliability and Validity of the Bioharness™ Multivariable Monitoring Device,” *Journal of Sports Science & Medicine*, vol. 11, no. 4, p. 643, 12 2012.

[25] P. Schmidt, A. Reiss, R. Duerichen, C. Marberger, and K. Van Laerhoven, “Introducing WESAD, a Multimodal Dataset for Wearable Stress and Affect Detection,” in *Proceedings of the 20th ACM International Conference on Multimodal Interaction*. New York, NY, USA: ACM, 2018.

[26] B. Zhao, Z. Wang, Z. Yu, and B. Guo, “EmotionSense: Emotion Recognition Based on Wearable Wristband,” in *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2018, pp. 346–355.

[27] D. Pollreisz and N. Taherinejad, “A simple algorithm for emotion recognition, using physiological signals of a smart watch,” *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, pp. 2353–2356, 9 2017.

[28] A. Albraikan, B. Hafidh, and A. El Saddik, “IAware: A Real-Time Emotional Biofeedback System Based on Physiological Signals,” *IEEE Access*, vol. 6, pp. 78 780–78 789, 2018.

[29] J. S. Heinisch, I. Hübener, and K. David, “The Impact of Physical Activities on the Physiological Response to Emotions,” *2018 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2018*, pp. 824–829, 10 2018.

[30] J. S. Heinisch, C. Anderson, and K. David, “Angry or Climbing Stairs? Towards Physiological Emotion Recognition in the Wild,” *2019 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2019*, pp. 486–491, 3 2019.

[31] D. Ferreira, V. Kostakos, and A. K. Dey, “AWARE: Mobile context instrumentation framework,” *Frontiers in ICT*, vol. 2, no. APR, p. 6, 4 2015.

[32] N. Naik, “Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP,” *2017 IEEE International Symposium on Systems Engineering, ISSE 2017 - Proceedings*, 10 2017.

[33] S. Bandyopadhyay and A. Bhattacharyya, “Lightweight Internet protocols for web enablement of sensors using constrained gateway devices,” *2013 International Conference on Computing, Networking and Communications, ICNC 2013*, pp. 334–340, 2013.

[34] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “ROS: an open-source Robot Operating System,” in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.