Mobile Proxemic Application Development for Smart Environments

Paulo Pérez
University of Pau /ESTIA Institute of Technology
LIUPPA – F-64210, Bidart, France
paulo.perez-daza@univ-pau.fr

Philippe Roose
E2S / University of Pau
LIUPPA – T2I64600, Anglet, France
philippe.roose@iutbayonne.univ-pau.fr

Yudith Cardinale
Universidad Simón Bolívar
Caracas, Venezuela, Universidad Católica San Pablo, Arequipa Perú
ycardinale@usb.ve

Mark Dalmau
E2S / University of Pau
LIUPPA – T2I64600, Anglet, France
dalmau@iutbayonne.univ-pau.fr

Dominique Masson
Technopole Izarbel Dev1-0
Bidart, France
d.masson@dev1-0.com

Nadine Couture
University of Bordeaux
ESTIA Institute of Technology, Bidart, France
n.couture@estia.fr

Abstract

Currently, mobile technologies are present in our daily day in different activities and their use keeps increasing. In this context, proxemic interaction, derived from proxemic theory, is becoming an influential approach to implement Mobile Human-Computer Interaction (MobileHCI) in smart environments, based on five proxemic dimensions: Distance, Identity, Location, Movement, and Orientation (DILMO). The existing tools for implementing proxemic applications require fixed devices that make it difficult to built mobile proxemic apps. This work aims to propose a framework for the development of proxemic applications for smart environments comprised by entities, whose interactions are supported by MobileHCI defined in terms of DILMO dimensions. Our proposed framework allows defining and managing all components in a smart proxemic environment. The framework also provides an API, that allows developers to simplify the process of proxemic information sensing (i.e., detection of DILMO dimensions) from mobile phones and wearable sensors. We demonstrate and evaluate the effectiveness and suitability of our framework, through the proof-of-concept which describes the implementation of two proxemic mobile applications built in a context-based infrastructure for smart proxemic environments based on mobile devices.

MoMM2020, 30 November - 2 December, 2020, Chiang Mai, Thailand

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

CCS Concepts • **Human-centered computing** → *User models*; *User interface programming*; Human computer interaction (HCI).

Keywords Proxemic interaction, proxemic zone, mobile devices, wearable technologies, MobileHCI.

ACM Reference Format:
Paulo Pérez, Philippe Roose, Yudith Cardinale, Mark Dalmau, Dominique Masson, and Nadine Couture. 2020. Mobile Proxemic Application Development for Smart Environments . In *MoMM2020: International Conference on Advances in Mobile Computing & Multimedia, 30 November - 2 December, 2020, Chiang Mai, Thailand.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

1 Introduction

Nowadays, the use of mobile technologies in our daily life is very common. People can interact with different contexts through electronic devices (e.g., personal mobile phones, tablets, wearable technologies, and smart-watches) to accomplish their daily tasks. Many of these tasks require a specific Human-Computer Interaction (HCI). Researchers are therefore seeking to develop new useful and enjoyable interfaces. Proxemic interaction arises as a novel concept to improve HCI [5, 16]. Proxemic interaction describes how people use interpersonal distances to interact with digital devices [3, 14, 15], using the so called five physical proxemic dimensions: Distance, Identity, Location, Movement, and Orientation (DILMO).

Proxemic interaction is derived from the social proxemic theory proposed in 1966 by the anthropologist Edward T. Hall [18]. Hall describes how individuals perceive their personal space relative to the distance between themselves and others. According to Hall's proxemic theory, interaction zones have been classified into four zones: (i) intimate zone, comprised between 0 and 50 cm of distance; (ii) personal zone, defined by a distance of 50 cm to 1 m; (iii) social zone, when the distance is between 1 m and 4 m; and (iv) public

zone, for distances of more than 4 m. He underlines the role of proxemic relationships as a method of communication based on the distance among people. This theory has been applied to define relation and communication among people and digital devices [14].

In this context, solutions such as Toolkit [23] and ProximiThings [6] (for proxemic interaction in the Internet of Things) have been proposed to support the development of proxemic interaction. However, existing tools and frameworks present limitations for implementing proxemic interaction in mobile technologies because they require special hardware devices connected to the system (e.g., a Kinect Depth sensor, which must be installed on a PC for sensing proxemic information).

Nowadays, the vast majority of smartphones and mobile devices are equipped with powerful hardware capabilities. These capabilities allow devices to process and obtain proxemic information; for example by using sensors and cameras embedded in a smartphone. In turn, it is possible to implement proxemic based mobile applications that facilitate users' contact and interaction with other people and devices in indoor and outdoor spaces, which we call smart proxemic environments. This fact, combined with the current trend of using proxemic interaction to improve HCI, has raised the need for frameworks and tools to support the development of such as mobile proxemic applications.

This work aims to propose a concrete solution, a framework, for developing mobile proxemic applications comprised by entities, whose interactions are defined according to DILMO dimensions Our proposed framework represents a threefold contribution: (i) it proposes a process to define and manage all components in a proxemic environment: the interaction objects, the DILMO dimensions that govern the HCI, and the proxemic mobile applications; (ii) an API integrated into the framework, that allows developers to simplify the process of proxemic information sensing (i.e., measure of DILMO dimensions) by mobile phones and wearable sensors; and (iii) the proof-of-concept to demonstrate and evaluate the effectiveness and suitability of our framework by describing the implementation of two mobile proxemic applications; these two mobile apps are based on HCI defined as a function of different DILMO combinations that specify different context-based infrastructures for proxemic environments based on mobile devices.

2 Related work

Proxemic concepts are based on physical, social, and cultural factors that influence and regulate interpersonal interactions [23]. In order to know how the factors should be applied to proxemic interactions for ubiquitous computing applications, Greenberg et al. [14] identified five dimensions: Distance, Identity, Location, Movement, and Orientation (we call them DILMO as an abbreviation), which are associated with people, digital devices, and non digital things. In this section, we review prior works on proxemic interaction and how they have been implemented. Afterward, we analyze the existing technical methods for the development of proxemic applications and compare them with our framework.

2.1 Applications based on Proxemic Interactions

There exist a variety of works that implement interactive ubiquitous applications. The common aspect to all these applications is the use of all or a subset of proxemic dimensions (i.e., DILMO). These dimensions allow applications to know absolute and relative positions of people and objects in the physical space. In this section, we describe the concept of each proxemic dimension and how they have been used by prior works.

Distance is a physical measure of separation between two entities, according to how they interact [24]. Typically, short distances allow high interactions, while long distances allow little to no interaction. For example, in [9, 10, 13, 16, 21, 23, 27, 34], the distance is used as a parameter to assign a proxemic zone that allows the users to interact with the display or devices in different proximities. The interaction zones are also used for adapting visualizations on displays based on the users' distance relative to the screen, such as the studies presented in [4, 19, 32].

Distance can be obtained by using different techniques based on a variety of sensors to capture their values. Bluetooth Low Energy (BLE) technologies allow the device to estimate the proximity among entities by Received Signal Strength Indicator (RSSI) and Broadcasting Power value (TX power), as in [35, 36]. The work presented in [7] uses a smartphone with BLE technology in order to obtain proximity between blind persons and fixed objects. The work presented in [21], proposes the use of the body-tracking capabilities of Kinect Sensors to obtain the distance. Authors demonstrate the suitability of their proposal in an application that measures the distance between blind people and paintings, according to which it provides different background music experiences. In [4, 25], computer vision is used for measuring the distance from the device hosting the program to the user.

Identity is a term that mainly describes the individuality or role of a person or a particular object in a space [3, 23]. SpiderEyes [10] is a collaborative proxemic system that helps designers to create applications by tracking multiple people interacting in front of a display in run-time. In this particular regard, it is indispensable to have the user identification. User's interaction is based on their identity and distances with the display. The system is able to detect when users leave the field of view of the display and if they later rejoin the field of view at a different distance. With SpiderEyes, authors demonstrate the effectivity of the identification system with up to four users at the same time. This work uses a visual monitoring tool (called Microsoft Kinect Depth Camera), that allows developers to classify which entities are being tracked and how the details of information are stored

in the application for creating identities. FAMA (First Aid Mobile Application) [29] is a mobile app based on proxemic interactions that offers rescuers to obtain emergency identification (identity) of an unconscious person, as the rescuers are moving toward the injured person's proxemic zones. FAMA uses Beacon BLE technology for the identification of entities (injured people). Proxemic-Aware Controls system [22] uses identity for controlling spatial interactions between a person's handheld device and all contiguous appliances to generate an effective appliance control interface. These applications have demonstrated that identity can be used to know individuality of a person or an object.

Location describes qualitative aspects of the space, where there is interaction among fixed entities (e.g., room layout, doors) or semi-fixed entities (e.g., furniture positions) [13, 22]. Researchers have also considered the location for obtaining the user's current positions. Multi-Room Music System [31] is based on proxemic interactions that allow the user to hear the same songs playlist, while he changes his location around the house through the speakers that have been installed in the rooms. In the work presented in [21], location is used to detect events related to hands' tracking. For example, when a blind user explores a painting with his hands, the application uses the 3D coordinate system of the Microsoft Kinect Camera [26], to know the user's hand position in a specific region of the painting. In [16, 23], entities are associated with three-dimensional positions related to a fixed point that can be used for initial setting of smart environments. In such a way, it is possible to obtain the relative position among people and devices.

Movement is defined as changes of position and orientation of an entity over the time [3]. This is the case when a user walks in front of a screen or approaches it, and the content of the screen is adjusted according to the user's movements. This kind of motion can be captured by motion technology [16, 33]. Velocity changes are calculated in order to respond to the user's behavior. The movement also allow gesture recognition through smartphones or wearable technologies by employing motion sensors [2]. FAMA, a first aid mobile app, identifies potential rescuers as they move towards the injured person's proxemic zones [29].

Orientation provides the information related to the direction in which an entity is facing. It can identify the front of an entity (e.g., person's eyes, screen front). Previous work have demonstrated how a person's orientation related to a display can be used for improving user interaction [3, 16, 24, 28]. The study presented in [17], describes the use of built-in compass in mobile devices to support the process of pairing them based on the orientation. Another remarkable work is Multi-View Proxemic system [11], which considers distinct views from a single display related to the angle of orientation of two viewers. This work uses gaze detection technology that allows the active user to be identified.

We have briefly described studies that have implemented multiple proxemic applications based on DILMO. From this review, we want to highlight that the majority of these proposals manage only partial proxemic dimensions; few of them have used the five DILMO proxemic dimensions [6, 13, 14, 20, 22, 23]. In all of these applications, the interaction objects (i.e., people and devices) are considered as entities; when the Identity dimension is implemented, these entities are explicitly identified in a particular role. For example, FAMA [29] uses a combination of Distance and Movement of people with respect to an identifiable person. In other studies, entities have been implicitly managed. For example in [9], Distance, Movement, and Location dimensions have been used to implement interaction between *a user* and *a screen*; the *user* and the *screen* are (non identifiable) entities. The work presented in [11] detects the Distance and Orientation of *a user* with respect to a *single display* to generate multiple views of the information displayed. Similarly, the applications described in [13, 21] help *visually impaired people* to explore paintings based on Distance, Movement, and Location. Thus, we conclude that depending on the application, all DILMO dimensions are not required and specific combination of them can determine different proxemic environments.

In next section, we present some works that have proposed tools and frameworks to support the development of applications based on proxemic interactions. We highlight their limitations and how we overcome them.

2.2 Tools to Develop Proxemic-based Applications

Some previous works have proposed tools to support the development of proxemic applications considering proxemic interactions. The work presented in [6], illustrates how the proxemic dimensions can support interaction among entities (people or objects), with a proposed context-aware framework. This framework provides capabilities that help developers build a front-end application. However, the framework requires a cloud computing architecture or an active connection to the server for processing proxemic information. There are applications in the medical field (neurosurgery) [25], based on proxemic interactions, where the offline state allows users to interact with the application without Internet access or an active connection to the local server. Therefore, we propose a framework that allows mobile devices to process all proxemic dimensions independent from server connections.

In [23], a framework, called Proximity Toolkit, used to discover novel proxemic-aware interaction techniques is proposed. The framework is a guide on how to apply proxemic interaction design for domestic ubiquitous computing environments. It is a collection of libraries developed in C and an architecture of components that make use of spatial information and relations among objects and space. This framework allows the rapid building of proxemic-aware systems and it

offers a flexible architecture for sensing proxemic data from different types of sensors. However, the implementation of this framework requires a hardware architecture based on fixed devices (e.g., a Kinect Depth sensor and a client-server architecture) for allowing the server to process the proxemic information from appliances. This solution does not offer the mobility and portability required for implementing proxemic interactions on mobile devices or wearable technologies [2]. With the current Proximity Toolkit version, it is not possible to obtain proxemic information from the new smartphone's sensors capabilities and ensuring that users can use proxemic mobile applications on their smartphones in any place.

All these works demonstrate the current interest for researchers to develop tools that support the design and implementation of proxemic applications. Proxemic interaction is a remarkable interaction technique that allows the user to control the digital devices in a flexible way [3, 22, 23]. Nowadays, smartphones and mobile technologies are powerful and offer a wide range of possibilities to improve user interaction. There are about 2 billion people with smartphones, which represents one-quarter of the global population in the world, and this trend is on the rise [12]. It is therefore of great importance to provide tools for developing mobile apps and improve HCI on mobile devices. However, proxemic interaction on mobile devices has not been implemented in full. There are some mobile applications based on proxemic interaction [4, 29, 30] that have been recently developed without the use of proxemic frameworks. Accordingly, to overcome those limitations and offer a solution for defining and managing smart proxemic environments, we propose a framework, which defines a systematic development process supported on an API, focused on creating proxemic applications using the smart-mobile sensors to obtain DILMO dimensions.

3 Proxemic Definitions

In this section, we present the definitions of proxemic environment and its components (i.e., entities, DILMO dimensions, and proxemic zones) that we have adapted from the proxemic studies.

The first term that we need to state is related to the objects that can interact in a proxemic environment. Def. 3.1 describes what we consider as interaction objects.

Definition 3.1. Entity (*E*). *An entity, denoted as E, represents an interaction object (e.g., a person, an object, a device), that can be univocally identified or not in a physical space.*

We keep the same significance of DILMO dimensions. Def. 3.2, Def. 3.3, Def. 3.4, Def. 3.5, and Def. 3.6 show our adapted definitions for **Distance**, **Identity**, **Location**, **Movement**, and **Orientation**, respectively.

Definition 3.2. Distance (*D*). *The distance, denoted as D, is the physical measure of proximity among mobile or fixed entities (E) in a physical space.*

Definition 3.3. Identity (*I*). *The Identity, denoted as I, represents an entity (E) that has a unique identification or a specific role in a physical space.*

Definition 3.4. Location (*L*). *The location, denoted as L, is a relative position or an absolute position of an entity (E) in a physical space.*

Definition 3.5. Movement (*M*). *The movement, denoted as M, represents the change in measures of distance or position of an entity (E), over an interval of time in a physical space.*

Definition 3.6. Orientation (*O*). *The orientation, denoted as O, represents the face to face alignment between two entities in a physical space.*

The physical space in which entities can interact according to DILMO variables is called a proxemic environment (see Def. 3.7).

Definition 3.7. Proxemic Environment (*P_E*). *A proxemic environment, denoted as P_E, represents a set of sensors and devices attached to entities (E) that can in turn interact according to D, I, L, M, and O.*

In the context of a proxemic environment (*P_E*), entities delimit proxemic zones, according to the distance *D*, between them. Def. 3.8 presents the formal meaning of a proxemic zone.

Definition 3.8. Proxemic Zone (*P_Z*). *A proxemic zone, denoted as P_Z, defines the proximity between two entities (E_1, E_2), according to a distance D, provided as a parameter. There are four P_Z: $P\_Z_{intimate}$, $P\_Z_{personal}$ P_Z_{social}, and P_Z_{public}*

In this work, we have defined four proxemic zones in which interactions among entities can be different. The distance that defines each *P_Z* is defined by developers according to user requirements.

These definitions conform the basis of our approach, implemented as a development framework. This framework is described in the next section.

4 Framework to Develop Proxemic Mobile Apps

We propose a framework, along with a simple systematic development approach, for supporting the construction of mobile proxemic apps for smart proxemic environments (*P_E*), based on mobile technology and smart wearable technology. The systematic development process provides a guidance on how to choose DILMO combinations for developing mobile apps, with particular MobileHCI for specific proxemic environments.

4.1 Framework Architecture

The framework architecture is designed to allow the developer to focus on how to obtain the sensing data from the

smart environment (i.e., smartphone, wearable device sensors. The contribution provided by the framework is to take advantage of current mobile technology trends related to the capabilities of gathering and processing different types of data that can be used to create proxemic applications. Our approach helps the developer with the development process and with the management of proxemic information to establish the appropriate combination of DILMO dimensions.

In order to create a mobile proxemic app and define a *P_E*, we propose a sequential process comprised by three single steps:

1. Define the proxemic zones (*P_Z*) according to which entities will interact.
2. Define the appropriate DILMO combination to define the MobileHCI for the target mobile app to be developed.
3. Implement the mobile app, considering the technology supported by the entities in the *P_E*.

To support this process, the framework is composed by mainly three components aligned with each step (see Figure 1): (i) Proxemic Zones module; (ii) DILMO module; and (iii) an API that supports the instantiation of the two previous mentioned components. In the following, we describe each module in detail.

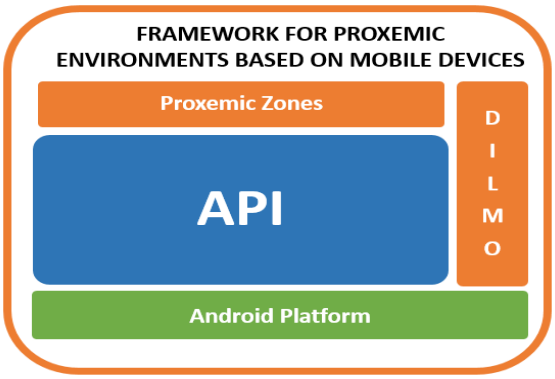

Figure 1. Framework architecture.

The **Proxemic Zones module** allows the definition of proxemic zones sizes (*P_Z*), according to user needs. The interaction between two entities changes in accordance of the *P_Z* in which they are located. These values of distances that delimit the proxemic zones are configured through the API.

Figure 2 is a guide that allows the developer to know which methods must be implemented on the API or which objects must be created from the API according to DILMO dimensions for processing proxemic information. For example, a DIL proxemic environment (row 10 in Figure 2) means that Distance (*D*) and Location (*L*) are considered for Identities (*I*).

Combination of proxemic dimensions are also valid, although the entities have not unique identification; e.g., *a person, a device beacon*, instead of *the smartphone's owner, my device beacon*. Hence, proxemic environments denoted in rows 10 to 20 in Figure 2, can become `DL`, `DM`, `DO`, `LM`, `LO`, `MO`, `DLM`, `DLO`, `DMO`, `LMO`, and `DLMO`, respectively, when all interaction objects are not identifiable entities.

The **API** facilitates developers processing proxemic information and values. The API provides classes and methods to define the *P_Z*, as well as to manage the different combinations of DILMO dimensions. For example, for a DIL proxemic environment, methods to identify entities (*I*) and to process *D* and *L* are available in DILMO class. Thus, the API behaves as a bridge between the Proxemic Zones and DILMO modules.

In the current version of our framework, the API considers the extraction of DILMO values from smartphones or mobile devices based on the Android native libraries (APKs). The API provides methods that the developers can implement for processing proxemic information using motion sensors and mobile computer vision cameras. The majority of current smartphones have a wide range of sensors in their hardware configuration [8], which allow the application to run proxemic apps. For example, through the BLE beacons mechanisms [1], it is possible to know the distance (*D*) between two mobile devices. Another way to estimate the distance between two entities is to use computer vision (face detection).

4.2 API Implementation

In this section, we describe the API structure and some of the most important available methods. The API lets developers build *P_Z* and process proxemic information from Android sensors, that are required for implementing a *P_E*. It was developed in Java, hence the `jar` files are provided to be added to the Android Studio platform. Figure 3 describes the structure of the API, represented by a UML Class diagram. The main classes are described as follows. The API[1] is available for free downloading.

1. The `ProxZone class` allows to define proxemic zones (*P_Z*) according to user requirements (i.e., user/developer decides the measures that delimit each *P_Z*), when this class is instantiated (i.e., by its constructor method). The constructor method of this class receives as parameters the respective maximum measures of distance *D*, which define each *P_Z*. A *P_Z* can be associated to one or more entities.
   Figure 4 shows an example of the `ProxZone` constructor method, in which the maximum distance, in meters, for each *P_Z* are specified (see Def. 3.8): $P\_Z_{intimate}$ is delimited from 0 to 0.25 meter, $P\_Z_{personal}$ is defined from 0.26 meter to 0.45 meter, P_Z_{social} is from 0.46 meter to 1 meter, and P_Z_{public} is depicted from 1.1

[1]The API is available in https://www.iutbayonne.univ-pau.fr/~ppdaza/

	D	I	L	M	O	MIX	Proxemic Environment
1	■					D	Physical length (D) between entities.
2		■				I	Identifiable (I) entities in a specific role in the interaction space.
3			■			L	Position (L) of an interaction object (entity).
4				■		M	Motion (M) capture of an interaction object (entity).
5					■	O	Face orientation (O) between two entities.
6	■	■				DI	Interaction based on proximity (D) between identifiable (I) entities.
7		■	■			IL	Interaction based on the physical location (L) of identifiable (I) entities.
8		■		■		IM	Interaction based on movement tracking (M) of identifiable (I) entities.
9		■			■	IO	Interaction based on face to face orientation (O) of identifiable (I) entities.
10	■	■	■			DIL	Interaction based on proximity (D) and positions (L) of identifiable (I) entities.
11	■	■		■		DIM	Interaction based on proximity (D) according to movement tracking (M) of identifiable (I) entities.
12	■	■			■	DIO	Interaction based on proximity (D) and face to face orientation (O) of identifiable (I) entities.
13		■	■	■		ILM	Interaction based on physical location (L) and movement tracking (M) of identifiable (I) entities.
14		■	■		■	ILO	Interaction based on face to face orientation (O) and position (L) of identifiable (I) entities.
15		■		■	■	IMO	Interaction based on movement tracking (M) and face to face orientation (O) of identifiable (I) entities.
16	■	■	■	■		DILM	Interaction based on proximity (D) and physical location (L) with movement tracking (M) of identifiable (I) entities.
17	■	■	■		■	DILO	Interaction based on proximity (D), location (L) and face to face orientation (O) of identifiable (I) entities.
18	■	■		■	■	DIMO	Interaction based on proximity (D) and face to face orientation (O) according to movement (M) of identifiable (I) entities.
19		■	■	■	■	ILMO	Interaction based on location (L) and face to face orientation (O) according to movement (M) of identifiable (I) entities.
20	■	■	■	■	■	DILMO	Interaction based on all proxemic interaction dimensions.

Figure 2. DILMO proxemic dimensions and nomenclature to describe each combination that is available through the API methods for processing proxemic information.

meters to 2 meters. Inappropriate arguments are validated in order to have valid measurements (e.g., not overlapped zones, right order of measures).

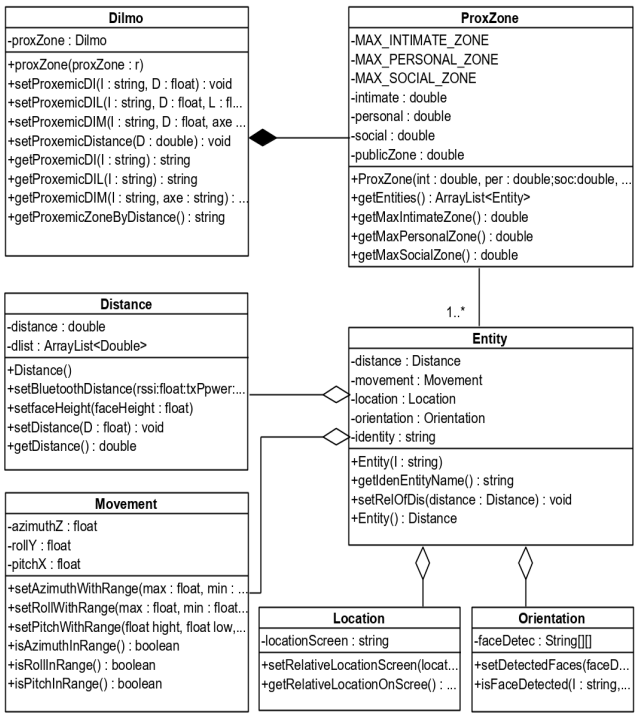

Figure 3. UML Class Diagram of the API.

2. The `DILMO class` is useful for developing *P_E*. It offers the possibility of identifying the *P_Z* of all entities *E* (or identities *I*) which will interact in the *P_E*. This class allows to define relations among proxemic dimensions, according to our proposed combinations of DILMO (see Figure 2). Its main methods are:
   a. The `setProxemicDI(String` *I*`, double D)` method allows assigning a *P_Z* to an identity (*I*), based on the distance (*D*).
   b. The `getProxemicDI(String` *I*`)` method allows obtaining the *P_Z* of an identity (*I*).
   c. The `setProxemicDIL(String` *I*`,double` *D*`, float` *L*`)` allows assigning a *P_Z* to an identity (*I*) based on the distance (*D*) and processing the location (*L*).
   d. The `getProxemicDIL(String` *I*`)` method allows obtaining the *P_Z* and relative location of the identity (*I*).
   e. The `setProxemicDistance(double D)` method allows assigning the *P_Z* to an entity, according to the distance (*D*).
   f. The `getProxemicZoneByDistance()` method returns the *P_Z* of an entity, based on the distance.
3. The `Distance class` allows the developer to estimate the distance among identities (*I*). Distance (*D*) can be calculated by using any available method. In the current version of our API, we have integrated some methods to calculate *D*, based on the Android platform, such as:

```
//Définition des zones proxémiques utilisée
proxzone = new ProxZone(0.25D, 0.45D, 1.0D, 2.0D)
```

Figure 4. Example of ProxZone Class Constructor invocation.

a. The `setBluetoothDistance(double rssi, double txtPower)` that allows estimating distance based on BLE.
b. The `setFaceHeight(float faceHeight)`, which allows estimating the distance from camera using visual computing; distance is proportional to the height of the detected face.
c. The `getDistance()` method, that allows obtaining D in meters.

4. The `Entity class` represents the interaction objects in a P_E, whose behavior is determined or will be determined according to their DILMO proxemic dimensions. This class allows the discovering of the entities on a P_E.
5. The `Location class` provides methods to manage location (L) of interaction objects (E and I). As in the `Distance class`, location (L) of entities can be calculated by any available method. Currently, we have integrated in our API some methods to calculate L, such as:
   a. The `setRelativeLocationScreen(float L)` method, which sets the relative position on the screen of an entity E, based on the coordinates of E on the display.
   b. The `getRelativeLocationOnScreen()` method, which returns the relative position of an entity E.
6. The `Movement class` has methods that allow motion processing from the coordinate system of smartmobile sensors (e.g., Azimuth, Pitch, Roll), such as:
   a. The `setAzimuthWithRange(float MAX, float MIN, float value)` method which allows processing the azimuth angle of an entity E, that has been established by the developer.
   b. The `isAzimuthInRange()` method returns true if the azimuth angle of an interaction object (E) is within a range of reference.
7. The `Orientation class` provides methods to validate the face orientation (O) of interaction objects on a P_E. Some of them are:
   a. The `setDetectedFaces(String[][] detectedFaces,ProxZone p)` method, that receives a collection of faces to be defined as interaction objects (E or I) in the P_E.
   b. The `isFaceDetected(String I,String P_Z)` method, which returns true if a specific face (I) is detected in the P_Z.

5 Proof-Of-concept Of our Framework

Our goal is to create proxemic environments based on mobile devices or wearable technology and demonstrate that our framework allows developers to build proxemic mobile applications effectively. For this purpose, we show the implementation of two mobile applications, called IntelliPlayer and Tonic, based on proxemic interactions. These apps were implemented using Android Studio platform version 3.3; however a higher Android studio version can be used. The apps[2] are available for free downloading

Both apps have been developed by undergraduate students, as part of their final project in computer science. The developing team was integrated by four students whose average age was 21 years-old, who have developer experience using Java object-oriented programming. This project was the first challenge for them implementing Android applications and MobileHCI based on proxemic interactions.

Two training sessions of two hours each were organised for the student. The training process allows students to understand the systematic process for building proxemic mobile applications with our framework. They learned: (i) how to define each P_Z; (ii) how to select each combination of proxemic dimension for recreating a P_E; and (iii) how to use methods and classes in the API. The development time of both applications was 64 hours by two developers over a period of 4 weeks. IntelliPlayer took 44 hours of work, while Tonic was finished in 20 hours, in the same four weeks.

IntelliPlayer is a mobile application that plays a video in a smartphone and reacts according to four proxemic zones and DILO proxemic dimensions. In the first step of the approach, the four P_Z were created: P_Z_{intime} (0 mts to 0.25mts), $P\_Z_{personal}$ (0.26 mts to 0.45 mts), P_Z_{social} (0.46mts to 1 mts), and P_Z_{public} (1.1 mts to 2 mts). Then, in the second step, the MobileHCI was designed according to D, I, L, and O, thus a DILO P_E was defined. Figure 5 shows the proxemic zones that have been defined by the developer through the API, as shown in Figure 4.

Figure 5. IntelliPlayer proxemic zones.

With this application, we illustrate a proxemic environment using a mobile player app that reacts to the distance (D) and location (L) of a person (E_1) and his face orientation (O), with respect to the smartphone (I_1) displaying a video. The computer vision technique has been used for this purpose, based on the properties of an Android camera and through the API methods `setFaceHeight(float faceHeight)` and `getDistance()` described respectively in items 3.(b) and 3.(c) in Section 4.2. Figure 6 shows a block code of this case.

With the distance (D) between the user (E_1) and the smartphone (I_1), IntelliPlayer determines the proxemic zone (P_Z)

[2]The APPS are available in https://https://github.com/llagar910e/

```
Distance d= new Distance();
d.setfaceHeight(faces.valueAt(i).getHeight());
String id =String.valueOf(faces.valueAt(i).getId());
dilmo.setProxemicDI(id, d.getDistance());
dilmo.getProxemicDI(id);
changerVolume(dilmo.getProxemicDI(id));
//Log.i("FaceId", "Zone : "+ id + "/"+ dilmo.getProxemicDI(id));
```

Figure 6. Block code of IntelliPlayer.

of E_1 (a user), with respect to the smartphone (I_1). To do so, it invokes the method `getProxemicZoneByDistance()` described in item 2.(f) in Section 4.2.

IntelliPlayer automatically adjusts the volume of the video according to the P_Z in which E_1 (the user) is with respect to the smartphone (I_1): when E_1 is in P_Z_{intime}, it decreases to 25% volume of speaker; for $P\_Z_{personal}$, it increases to 50% volume; for P_Z_{social}, it increases to 75% volume; and for P_Z_{public}, it increases to 100% volume) (see Figure 5).

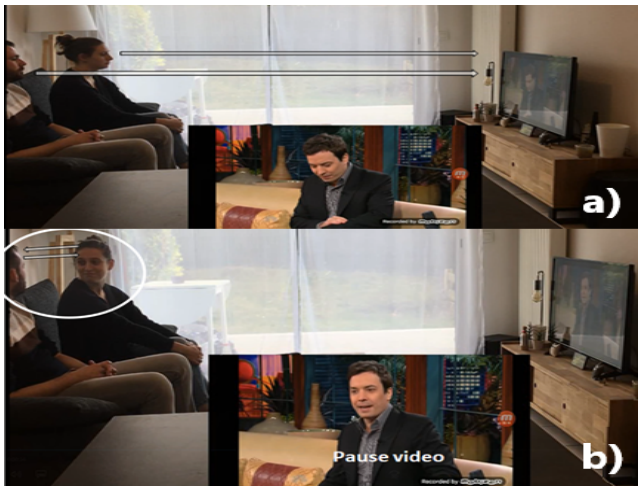

Figure 7. Play pause video using users faces orientation.

When a second person (E_2) is in front of the smartphone camera (I_1), the application verifies if both users are looking at the screen at the same time, as shown in Figure 7(a) (method `setDetectedFaces (String[][]detectedFaces, ProxZone p)`, item 7.(a) in Section 4.2). In the case one user (E_1 or E_2) turns his face, the video will be paused automatically by the mobile application (see Figure 7(b)).

Another useful function of IntelliPlayer is to provide a video description that users can read on the screen according to user location (L) (see Figure 8). When a user (E_1 or E_2) is in the $P\_Z_{personal}$ and his orientation (O) is in front of the screen, the application can obtain the face location (L) (see `setRelativeLocationScreen(float L)` and `getRelativeLocationOnScreen()` in item 5.(a) and 5.(b) in Section 4.2) to split the screen, with the video (running) and information about the video on the right or on the left, according to the detected L. The correct use and instance of methods and classes of the API conforms the third step of the proposed approach.

Figure 8. The split view provides video description.

Tonic is an educational mobile app for learning musical notes, developed for illustrative purposes. In the first step of the approach, the students have defined four proxemic zones: P_Z_{intime} (0 mts to 0.5mts), $P\_Z_{personal}$ (0.51 mts to 1 mts), P_Z_{social} (1.1 mts to 2 mts), and P_Z_{public} (2.1 mts to 4 mts). In the second step, a DIMO combination was stated for the proxemic environment, P_E.

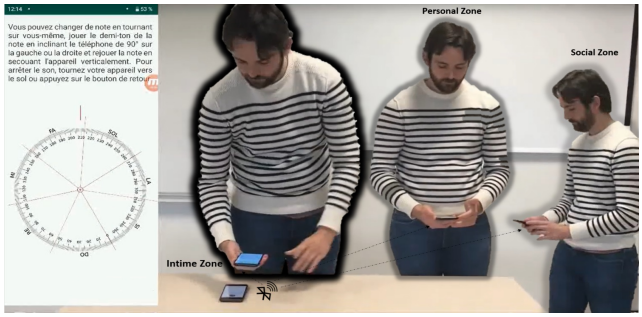

Figure 9. Tonic Proxemic Zones based on Bluetooth Low Energy.

Tonic allows a user to play a note and modify it from her/his smartphome on another smart mobile. The user's smartphone is identified (i.e., it is an I_1) by the mobile device which plays the sound (i.e., it is an I_2). Thus, I_2 plays and modifies a sound, by using proxemic interactions based on the P_Z, and on D, I, M, and O dimensions (i.e., a DIMO proxemic environment). In Tonic, the distance (D) between the two devices is obtained by using BLE technology. I_1 broadcasts its identifier to nearby portable electronic devices, thus it is caught by I_2. The volume of the sound is adjusted according to the P_Z in which I_2 is, with respect to I_1 (see Figure 9). The musical notes are changed according to the movement (M) and orientation (O) of I_2, with respect to I_1. According to M a tone is increased/decreased, while according to O a semi-tone is increased/decreased. M and O are calculated based on the capabilities of the smartphone, such as accelerometer, gyroscope, compass, and magnetometer. These sensors provide proxemic information that is mainly used in the API. Movement M was determined by using methods `setAzimuthWithRange(float MAX, float MIN, float value)` and `isAzimuthInRange()` described

in items 6.(a) and 6.(b) in Section 4.2; while *O* was managed by methods based on the smartphone's technical capacities. To manage *P_Z* and *D*, the methods used from the API, in the third step of the approach, were those described in items 2.(b), 2.(c), and 3.(a) in Section 4.2: `ProxemicDI(String I)`, `setProxemicDIL(String I ,double D, float L)`, and `setBluetoothDistance(double rssi, double txtPower)`.

In order to evaluate the usability of the API, we applied a survey composed of nine questions to the group of students who developed the applications described. Results of the survey are shown in Figure 10. These results indicate that 100% of surveyed students have strongly agreed with **Q1**: *"It was easy to implement the API with Android Studio"* and **Q9**: *"The API is useful for the development of proxemic applications with Android?"*. While 95% of students endorsed **Q5**: *"The API allowed you to process information of a combination of DILMO dimensions"*, **Q7**: *"The API's method for estimating distance based on face detection was accurate"*, and **Q8**: *"The API allows you to obtain the proxemic zone when it is using Bluetooth proximity sensing"*. For **Q4**: *"The API allows you to improve your productivity for developing proxemic applications by hiding complexity"* and **Q6** *"The API allows you to create the proxemic zones quickly"*, 78% of students expressed acceptation. Finally, **Q2**: *"The documentation provides enough information to interact with the API"* and **Q3**: *"The API provides enough examples for creating proxemic applications"*, were accepted only by 53% and 61% of students, respectively.

The the proof-of-concept and survey allow corroborating that the API supports development of proxemic applications and creation of proxemic environments, based on the exclusive use of mobile devices, while reducing complexity of the development process. These apps offer portable implementations that facilitate using the proxemic interaction in comparison to previous works that have used fixed platforms for similar purposes. Moreover, the survey results allow knowing aspects to develop, such as the quality of documentation and the basic examples provided.

6 Conclusion

Proxemic interaction provides different options for HCI and mobile interaction while offering several advantages and better experiences for users. Through this work, we have explored the use of proxemic interaction based on the combination of proxemic dimensions, called DILMO (Distance, Identity, Location, Movement, and Orientation) to offer a new context-oriented interaction for mobile applications. We have presented a systematic process to guide developers on creating realistic proxemic environments supported by an API and a framework, in which the end-users only need to use mobile or wearable devices. We are currently working on a definition of a domain-specific language (DSL) to design proxemic environments, as well as the continuing implementation testing of the API with students. With this framework, we hope to inspire other researchers to build more proxemic mobile applications using social distancing.

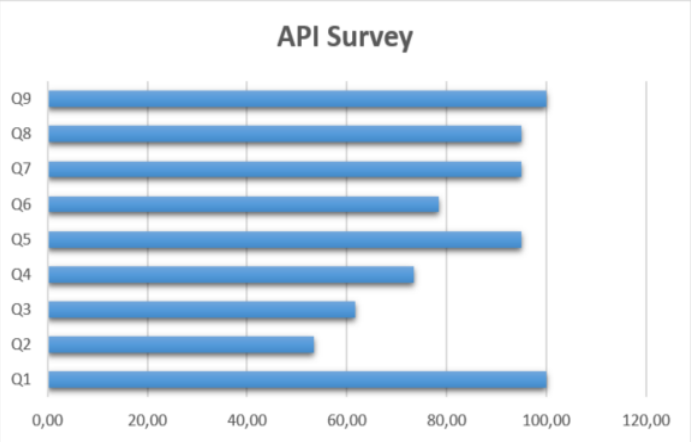

Figure 10. Results of students feedback.

Acknowledgments

This research was partially supported by Nouvelle Aquitaine, Communaute d'Agglomeration Pays Basque. We thank our students who contributed to the implementations of the mobile proxemic apps.

References

[1] AltBeacon. 2018. The Open and Interoperable Proximity Beacon Specification. https://altbeacon.org/.

[2] Mihai Bâce, Sander Staal, Gábor Sörös, and Giorgio Corbellini. 2017. Collocated Multi-user Gestural Interactions with Unmodified Wearable Devices. *Augmented Human Research* 2, 1 (2017), 6. http://doi.org/10.1007/s41133-017-0009-z

[3] Till Ballendat, Nicolai Marquardt, and Greenberg Saul. 2010. Proxemic interaction: designing for a proximity and orientation-aware environment. In *Proceedings of International Conference on Interactive Tabletops and Surfaces (ITS' 10)*. ACM, 121–130. http://doi.org/10.1145/1936652.1936676

[4] Michael Brock, Aaron Quigley, and Per Ola Kristensson. 2018. Change blindness in proximity-aware mobile interfaces. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, 43. https://doi.org/10.1145/3173574.3173617

[5] Frederik Brudy, Christian Holz, Roman Rädle, Chi-Jui Wu, Steven Houben, Clemens Nylandsted Klokmose, and Nicolai Marquardt. 2019. Cross-Device Taxonomy: Survey, Opportunities and Challenges of Interactions Spanning Across Multiple Devices. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, 562. https://doi.org/10.1145/3290605.3300792

[6] Carlos Cardenas and J Antonio Garcia-Macias. 2017. ProximiThings: Implementing Proxemic Interactions in the Internet of Things. *Procedia Computer Science* 113 (2017), 49–56. https://doi.org/10.1016/j.procs.2017.08.286

[7] Seyed Ali Cheraghi, Vinod Namboodiri, and Laura Walker. 2017. GuideBeacon: Beacon-based indoor wayfinding for the blind, visually impaired, and disoriented. In *Proceedings of International Conference on Pervasive Computing and Communications (PerCom '17)*. IEEE, 121–130. https://doi.org/10.1109/PERCOM.2017.7917858

[8] Google Developers. 2109. Sensors Overview. https://developer.android.com/guide/topics/sensors/sensors_overview#java.

[9] Tilman Dingler, Markus Funk, and Florian Alt. 2015. Interaction proxemics: Combining physical spaces for seamless gesture interaction. In *Proceedings of the 4th International Symposium on Pervasive Displays (PerDis '15)*. ACM, 107–114. https://doi.org/10.1145/2757710.2757722

[10] Jakub Dostal, Uta Hinrichs, Per Ola Kristensson, and Aaron Quigley. 2014. SpiderEyes: designing attention-and proximity-aware collaborative interfaces for wall-sized displays. In *Proceedings of the 19th international conference on Intelligent User Interfaces (IUI '14)*. ACM, 143–152. https://doi.org/10.1145/2557500.2557541

[11] Jakub Dostal, Per Ola Kristensson, and Aaron Quigley. 2013. Multi-view proxemics: distance and position sensitive interaction. In *Proceedings of the 2nd ACM International Symposium on Pervasive Displays (PerDis '13)*. ACM, 1–6. https://doi.org/10.1145/2491568.2491570

[12] Lingling Gao, Kerem Aksel Waechter, and Xuesong Bai. 2015. Understanding consumers' continuance intention towards mobile purchase: A theoretical framework and empirical study–A case of China. *Computers in Human Behavior* 53 (2015), 249–262. https://doi.org/10.1080/0144929X.2013.789081

[13] J Antonio Garcia-Macias, Alberto G Ramos, Rogelio Hasimoto-Beltran, and Saul E Pomares Hernandez. 2019. Uasisi: a modular and adaptable wearable system to assist the visually impaired. *Procedia Computer Science* 151 (2019), 425–430. https://doi.org/10.1016/j.procs.2019.04.058

[14] Saul Greenberg, Nicolai Marquardt, Till Ballendat, Rob Diaz-Marino, and Miaosen Wang. 2011. Proxemic interactions: the new ubicomp? *Interactions* 18, 1 (2011), 42–50. https://doi.org/10.1145/1897239.1897250

[15] Jens Emil Grønbæk, Mille Skovhus Knudsen, Kenton O'Hara, Peter Gall Krogh, Jo Vermeulen, and Marianne Graves Petersen. 2020. Proxemics Beyond Proximity: Designing for Flexible Social Interaction Through Cross-Device Interaction. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–14.

[16] Jens Emil Grønbæk, Christine Linding, Anders Kromann, Thomas Fly Hylddal Jensen, and Marianne Graves Petersen. 2019. Proxemics Play: Exploring the Interplay between Mobile Devices and Interiors. In *Proceedings of Companion Publication of the Conference on Designing Interactive Systems (DIS '19)*. ACM, 177–181. https://https://doi.org/10.1145/3301019.3323886

[17] Jens Emil Grønbæk and Kenton O'Hara. 2016. Built-in device orientation sensors for ad-hoc pairing and spatial awareness. In *Proceedings of Cross-Surface Workshop*. https://doi.org/10.13140/RG.2.2.16304.20488

[18] Edward T Hall. 1966. *The Hidden Dimension: An anthropologist examines man's use of space in private and public*. New York: Anchor Books; Doubleday & Company, Inc.

[19] Mikkel R Jakobsen, Yonas Sahlemariam Haile, Søren Knudsen, and Kasper Hornbæk. 2013. Information visualization and proxemics: design opportunities and empirical findings. *IEEE transactions on visualization and computer graphics* 19, 12 (2013), 2386–2395. https://doi.org/10.1109/TVCG.2013.166

[20] Han-Jong Kim, Ju-Whan Kim, and Tek-Jin Nam. 2016. miniStudio: Designers' Tool for Prototyping Ubicomp Space with Interactive Miniature. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, 213–224. https://doi.org/10.1145/2858036.2858180

[21] Kyle Kyle, Keith Salmon, Dan Thornton, Neel Joshi, and Meredith Ringel Morris. 2017. Eyes-Free Art: Exploring Proxemic Audio Interfaces For Blind and Low Vision Art Engagement. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 3 (2017), 93. https://doi.org/10.1145/3130958

[22] David Ledo, Saul Greenberg, Nicolai Marquardt, and Sebastian Boring. 2015. Proxemic-aware controls: Designing remote controls for ubiquitous computing ecologies. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI'2015)*. ACM, 187–198.

[23] Nicolai Marquardt, Robert Diaz-Marino, Sebastian Boring, and Saul Greenberg. 2011. The proximity toolkit: prototyping proxemic interactions in ubiquitous computing ecologies. In *Proceedings of the 24th annual ACM symposium on User interface software and technology (UIST '11)*. ACM, 315–326. https://doi.org/10.1145/2047196.2047238

[24] Nicolai Marquardt, Ken Hinckley, and Saul Greenberg. 2012. Cross-device interaction via micro-mobility and f-formations. In *Proceedings of the 25th annual ACM symposium on User interface software and technology (UIST '12)*. ACM, 13–22. https://doi.org/10.1145/2380116.2380121

[25] Helena M Mentis, Kenton O'Hara, Abigail Sellen, and Rikin Trivedi. 2012. Interaction proxemics and image use in neurosurgery. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'2012)*. ACM, 927–936.

[26] Microsoft. 2109. CameraSpacePoint Structure. Microsoft Docs https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn758354(v%3Dieb.10).

[27] Ghare Mojgan, Pafla Marvin, Caroline Wong, James R Wallace, and Stacey D Scott. 2018. Increasing Passersby Engagement with Public Large Interactive Displays: A Study of Proxemics and Conation. In *Proceedings of the International Conference on Interactive Surfaces and Spaces (ISS'18)*. ACM, 19–32. https://doi.org/10.1145/3279778.3279789

[28] Ahmed E Mostafa, Saul Greenberg, Emilio Vital Brazil, Ehud Sharlin, and Mario C Sousa. 2013. Interacting with microseismic visualizations. In *Proceedings of CHI Extended Abstracts on Human Factors in Computing Systems (HRI'13)*. ACM, 1749–1754. http://doi.org/10.1145/2468356.2468670

[29] Paulo Pérez, Philippe Roose, Dalmau Marc, Nadine Couture, Yudith Cardinale, and Dominique Masson. 2018. Proxemics for First Aid to unconscious injured person. In *Proceedings of the 30th Conference on l'Interaction Homme-Machine (IHM'18)*. 156–162. https://doi.org/10.1145/3286689.3286712

[30] Henrik Sørensen and Jesper Kjeldskov. 2012. The interaction space of a multi-device, multi-user music experience. In *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design (NordiCHI '12)*. ACM, 504–513. https://doi.org/10.1145/2399016.2399094

[31] Henrik Sørensen, Mathies G Kristensen, Jesper Kjeldskov, and Mikael B Skov. 2013. Proxemic interaction in a multi-room music system. In *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration (OzCHI '13)*. ACM, 153–162. https://doi.org/10.1145/2541016.2541046

[32] Jo Vermeulen, Kris Luyten, Karin Coninx, Nicolai Marquardt, and Jon Bird. 2015. Proxemic flow: Dynamic peripheral floor visualizations for revealing and mediating large surface interactions. In *Proceedings of IFIP Conference on Human-Computer Interaction (INTERACT'15)*. Springer, 264–281. https://doi.org/10.1007/978-3-319-22723-8_22

[33] Vicon. 2108. About Vicon Motion Systems. https://www.vicon.com/vicon/about.

[34] Daniel Vogel and Ravin Balakrishnan. 2004. Interactive public ambient displays: transitioning from implicit to explicit, public to personal, interaction with multiple users. In *Proceedings of the 17th annual ACM symposium on User interface software and technology (UIST '04)*. ACM, 137–146. https://doi.org/10.1145/1029632.1029656

[35] Yapeng Wang, Xu Yang, Yutian Zhao, Yue Liu, and Laurie Cuthbert. 2013. Bluetooth positioning using RSSI and triangulation methods. In *Proceedings of the 10th Consumer Communications and Networking Conference (CCNC'13)*. IEEE, 837–842. https://doi.org/10.1109/CCNC.2013.6488558

[36] Augustin Zidek, Shyam Tailor, and Robert Harle. 2018. Bellrock: Anonymous Proximity Beacons From Personal Devices. In *Proceedings of International Conference on Pervasive Computing and Communications (PerCom'18)*. IEEE, 1–10. https://doi.org/10.1109/PERCOM.2018.8444603