Polistudio

Davide Pezzolla, Giuseppe Piscopo, Maurizio Marseguerra

1 Overview

This project provides a service to allow people to monitor study rooms in real time. The users connecting to a web page are able to understand how many seats are free examining a map showed by the client. Such a map represents the geometry of the room and it reflects the real position both of the seats and of the tables.

2 Architecture

The architecture includes some modules that work together. A graphic diagram is shown below:

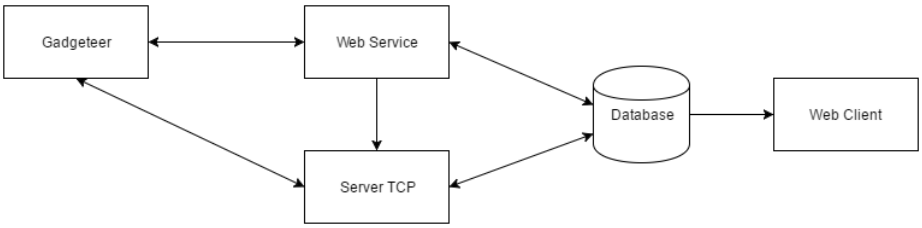

Figure 1: Architecture scheme

- Gadgeteer is the only one hardware module. Its task is to take a photo every n seconds and to send it to a serverTCP. The module communicates both with the web service and with the serverTCP.
- Web Service is a SOAP server based developed through WCF framework. It handles the authentication of the gadgeteer and instances a new serverTCP every time that a photo have to be received. In fact each photo transfer takes place in a new tcp connection like the ftp protocol.
- ServerTCP is a module that receives the photo, process it and update the database.
- Database takes care of store the data.
- Web client is the front end module. It uses a web socket that allow it to get the updated data in real time and refresh the graphics interface without any interaction by the user.

3 Gadgeteer

In this section we analyze what gadgeteer do. When the module is turned on, it begins an initial set up. First it try to connect with the network by using the ethernet module. When the connection is established, it connects and authenticates to web service. For the authentication phase it sends to web service a login request including your username and password, assigned for the specific study room. Then, if the user exists, receives a token that will use for the next requests. By contrast if user doesn't exist, it receives a null token and any other operation will can do. Once it receives a valid token, it sets a timer to know when it has to take a new photo. Then initial set up phase is completed, so gadgeteer waits that timer triggers. When happen, it takes a photo and sends to server.

The send of the image is composed of several stage. Since we wanted to avoid serializing of the image, we discarded the idea to send the image directly to web service. Serializing the image would have involved a slower transfer, so we adopt a solution like ftp protocol. Every time that a photo has to be sent, gadgeteer asks a port to web service. Once it is obtained, it creates a new tcp connection with the tcp server at that port, send the image and close the connection. In this manner we are able to send the image without serialize, but using directly socket class that allows to send a byte array throught the tcp connection.

Below we can see a time diagram that summarize gadgeteer operations:

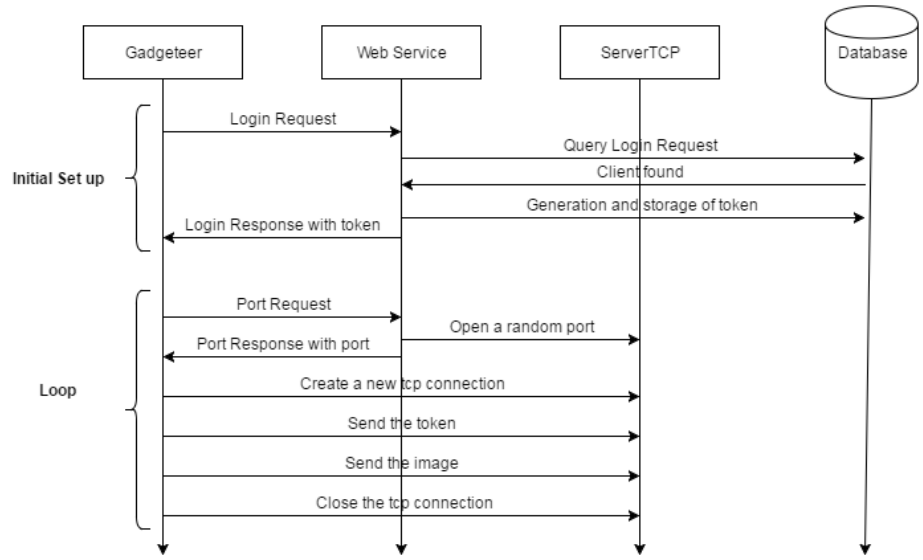

Figure 2: Time diagram

```
Connect to network
if( connected ){
   Login to Web Service through username and password
   if( logged ){
      a token is received
      Set timer for the photo
```

```
          every n seconds{
          Take a photo
          Request a port to web service using a token
          Create a tcp connection with the server at port
              obtained
          Send the photo
          Close tcp connection
           }
    }
    else{
       try to login for 3 times then need to be restarted
    }
}
```

Listing 1: Pseudo code of Gadgeteer operations

4 Web Service

The web service is realized by means of WCF (Windows Communication Foundation), which is a framework for building service-oriented applications. Such a framework has allowed us to focus on the service's developing providing tools for the generation of the proxy, (de)serialization, data validation and security management. We expose one endpoint to the client whose properties are:

Address https://192.168.1.2/gadgeteer/

Binding wsHttpBinding

Contract which describes the methods exposed to the clients, its description into WSDL file can be used to automatically generate a proxy by means of MFSvcUtil Windows tool.

```
[ServiceContract (Namespace = "http://polistudio/")]
public interface IService{
  [OperationContract]
  String login(String username, String password);

  [OperationContract]
  int portRequest(String token);

  [OperationContract]
  Boolean logout(String token);

}
```

Listing 2: Service contract

Web service paradigm is session-less, in order to keep trace of clients authentication we exploit a token, which is a random string generated by the login operation; only who submits a valid token is able to call the portRequest and logout exposed methods. Another problem is due to serialization poor performance, because of it pictures are sent through a TCP socket, portRequest is the operation in charge of opening a port for the client upon request. A final

consideration is about security, in order to ensure privacy, integrity and server authentication we use SOAP over https, the following listing shows the part of web.config in charge of encryption enabling.

```
<bindings>
  <wsHttpBinding>
    <binding name="wsHttpBindingNoSecurity">
      <security mode="Transport">
        <transport clientCredentialType="None" />
      </security>
    </binding>
  </wsHttpBinding>
</bindings>

<behaviors>
  <serviceBehaviors>
    <behavior name="ServiceBehaviour">
      <serviceMetadata httpsGetEnabled="true"
          httpGetEnabled="false"/>
      <serviceCredentials>
        <serviceCertificate findValue="polistudio"
            storeLocation="LocalMachine" storeName="My"
            x509FindType="FindBySubjectName"/>
      </serviceCredentials>
    </behavior>
  </serviceBehaviors>
</behaviors>
```

Listing 3: Web.config - security configuration

5 Server TCP

Server TCP is the software module that cover a central role in the project. For each photo transfer, a new TCP server is instanced by the web service. When it starts, it waits to receive a new photo. In addition to photo, it receives the token, in order to know from which study room the frame comes from. Once the photo is received and the room is identified, the server requests to the database the information about seats and tables. Then using a graphical algorithm process the image and updates the database with the new information.

6 Graphic processing

In order to process the images taken by the Gadgeteer we exploit the features of OpenCV, which is a multiplatform library with a strong focus on real-time applications. The information we want to obtain from the pictures has a double nature, on the one hand we need to know how many tables and seats are in the room and their position while on the other hand the number of people and where they are sit. The algorithm which computes such information is dived into 2 phases: learning and monitoring ones.

6.1 Learning phase

It determines the geometry of the room locating tables and seats; such a task is achieved by means of the comparison between room's picture and a "Frame0", which is a photo of the empty room.

Tables discovery is performed on Frame0 by means of a recognition algorithm able to identify shapes.

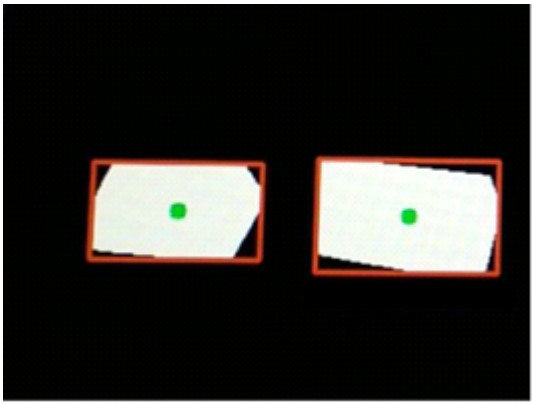

Figure 3: Shape recognition phase

Seats recognition is based on the identification of significant differences between a picture and Frame0, in fact the algorithm locates groups of points that do not match with Frame0 ones. In order to discover all the available spots of the room all the seats should be occupied at least one time in the analyzed frames, for this reason the learning phase could persist some days. However the mere frame comparison deceives the algorithm, in fact it could identify a walking person as an occupied seat. In order to avoid "ghost seat detection" we adopt a sampling technique: each discovered seat is associated to a counter which is incremented each time the seat is detected in a frame; after the end of the learning phase all the seats having the counter under a threshold are deleted from the seat list.

6.2 Monitoring phase

During the monitoring phase the algorithm stop to learn and it begin to assign a state to each seat, understanding if it is either free or occupied. The entire procedure is based on the recognition of BLOBs (Binary Large OBject), which are groups of connected pixels in a binary image. The term "Large" indicates that only objects of a certain size are of interest and that the other "small" binary objects are usually noise. The BLOB analysis is divided into 3 procedures:

1. BLOB extraction The purpose here is to isolate the BLOBs (objects) in a binary image. As mentioned above, a BLOB consists of a group of connected pixels. Whether or not two pixels are connected is defined by the connectivity, that is, which pixels are neighbours or not.

2. BLOB representation When you have extracted your BLOB, the next step is now to classify the different BLOBs. For example we want to classify

each BLOB as either a circle of not a circle, or a human vs. non-human BLOB. The classification process consists of two steps. First, each BLOB is represented by a number of characteristics, denoted as features, and second some matching method is applied to compare the features of each BLOB with the features of the type of object we are looking for. For example, to find circles we could calculate the circularity of each BLOB and compare that to the circularity of a perfect circle. So, BLOB representation is a matter of converting each BLOB into a few representative numbers. That is, keep the relevant information and ignore the rest. Such of information include for example, the area, the circularity, the compactness, the perimeter, the center of mass, etc...

3. BLOB classification The task here is to determine which BLOB for example, is a circle and which not. The question here now is how to define which BLOBs are circles and which are not based on their features that we mentioned earlier. For this purpose usually you need to make a prototype model of the object you are looking for. That means, what are the feature values of a perfect circle and what kind of deviation will you accept. This prototype model could differ from a simple box classifier (i.e. just a threshold value/limit) to a more advanced statistical classifier.

Following it is showed all the stages in which a picture is subjected during the graphical processing.

Difference It compares Frame0 (the empty room) with the "actual" frame received from the Gadgeteer, such an operation produces an image containing the objects that are not in Frame0.

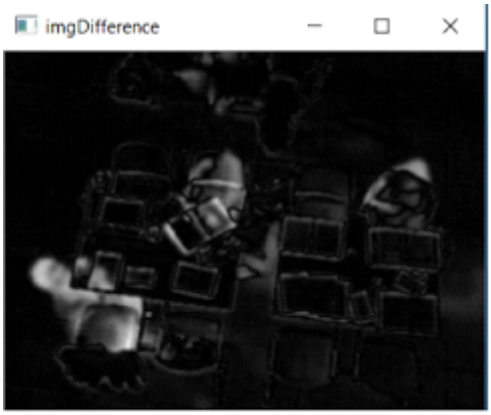

Figure 4: Difference phase

Thresholding The aim here is to highlight the differences founded in the previous step by means of a threshold. Such an action is achieved improving the definition of the pixels which allows to process the image on a given light intensity interval. In order to perform the needed operation some parameters has to be calibrated, such a calibration has been performed

manually during the test because they depend on the actual light condition. An automatic parameters configuration can be introduced with the deployment of a light sensor.

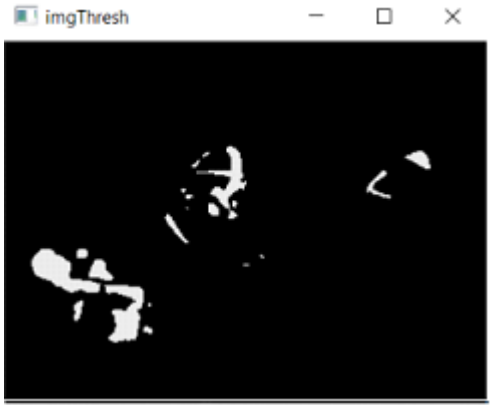

Figure 5: Thresholding phase

Erode/Dilate These steps expand and erode the neighboring pixels, in order to obtain a single object (person or object).

Contours detection Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition.

Figure 6: Contours detection phase

BLOB detection The detection is achieved by means of a simple algorithm which is driven by standard parameters that are used to define constraints that allow to understand if an object has the shape of a person (e.g. analyzing the subject area, the diagonal of the object etc.).

Figure 7: Blob detection phase

BLOB recognition Applying some contours we can finally define a set of spots.

Figure 8: Spots recognition output

Despite the effort to improve the algorithm it is possible that false positives are detected. Such a behavior is due to the position of the camera that is not perpendicular with the floor and increases the degree of imprecision of the system. For this reason it is possible that the presence of some objects which resemble humans (in terms of area, diagonal and other parameters) change the state of a seat. However thanks to some constraints defined into the algorithm, small objects do not affect the output of the processing.

7 Web Client

The web client is the module in charge of showing to the world the seat occupation states of the monitored room. Its valuable feature concerns the automatic view updating that is implemented exploiting web socket technology. Such a

technology is realized through a protocol providing full-duplex communication channels over a single TCP connection, in this way the client can receive data from the server without sending any request, so the interface is able to evolve without any interaction by the user.

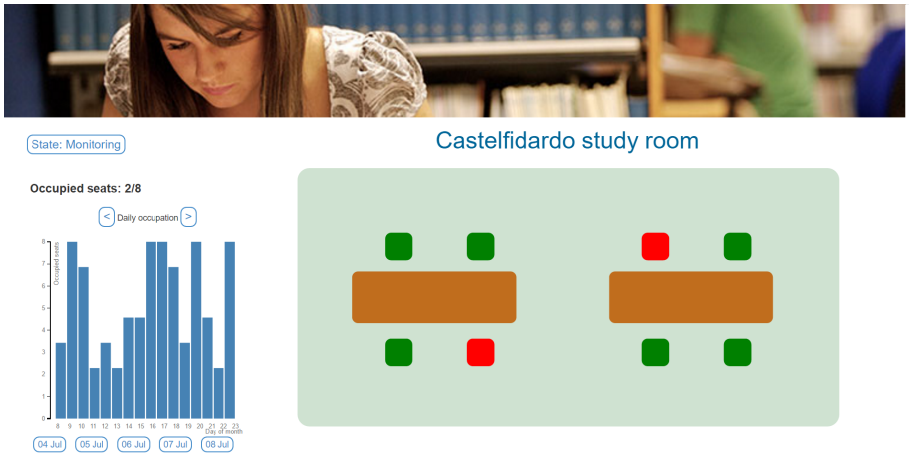

Figure 9: Web client interface

8 Security

In this project we wanted to take account of security.

To be able to send an image, gadgeteer has to be authenticated to web service. This authentication is based on username and password and it is necessary to identify the study room that gadgeteer is monitoring. However we wanted to avoid that someone might be able to get these information and performing any requests to web service pretend to be the gadgeteer. For this reason, we have insert an SSL encryption in the communication from the gadgeteer to web service, so no username and password will be visible through the network. However, since the authentication phase takes place only in the initial set-up, we need a way to keep the session. To do this we use a token. When a gadgeteer send a login request, the web service checks in the database if username and password are corrected. If they are, it generates a random token and maps the user (gadgeteer) with that token. Then, this is sent to module and it will be use in all the future communications.

Regarding the communication from gadgeteer and tcp server, we decided to avoid to use the SSL encryption. We tried to use, but photo transfer had become slower. However each photo is sent through a new tcp connection established on a different port. So even if someone is able to sniff the token, he can not know which is the open port where sending the images because this is exchanged through an encrypted communication and it is valid only for one transfer. For this reason, use unsecured connection for these communications is not as dramatic.