Medibot

+

Leaflet

Written by Jake Davis, Luke Bonner & Euan Gilfillan

Funded by Inside Pensions

The Priory School, Hitchin

Table of Contents

Project Overview and Aims

- Over 3.5 people over the age of 65 live alone
- According to Age UK, 50,000 use their pendant alarm system (see Figure 1 Traditional Pendant Alarm System).
- We do not think that the traditional alarm system is sufficient as when the alarm is activated the help centre cannot fully establish what assistance the person requires.

- To respond to this problem, we have created a robot to be kept in houses with elderly and disabled inhabitants to be used to give people independence in their homes.

- Medibot is designed to provide contact between vulnerable people and a help service, should they require assistance.
- Medibot works as a camera between the call centre and the client, the employee at the call centre would be able to control the device, be able to assess the situation and execute the correct protocol.

- Our idea originated when we saw a weakness in the panic button-style system.
- When the button is pressed, a person is notified and must make a call on further action with no information at all from the person's current circumstances

- We were motivated to do the project due to the recent NHS budget cuts. This meant that fewer people could be live in full time accommodation at care homes and would therefore have to live independently.
- Our project would be able to provide safety to these people and to help people if they are ever in danger.

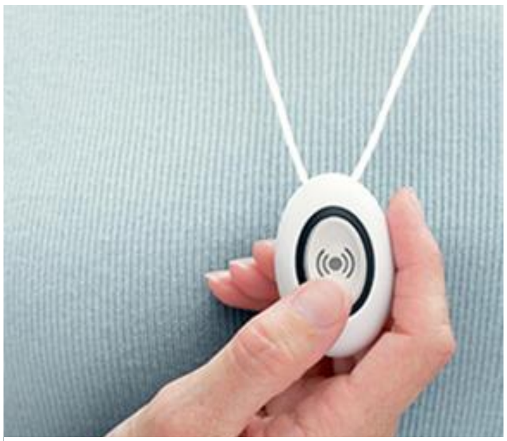


Figure 1 Traditional Pendant Alarm System

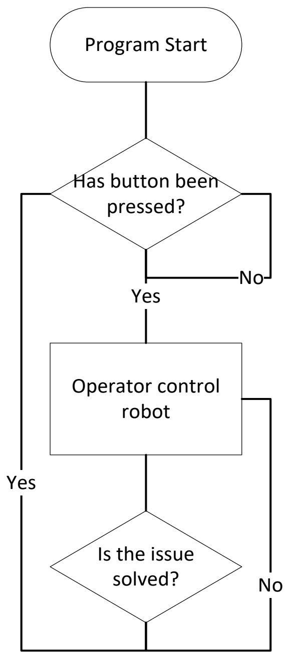
Design

There are three key elements of our project which we aimed to complete whilst developing the robot:

- Robustness
    - If it was deployed to people's homes, the robot must always work.
    - It is central that the robot is reliable and has effective redundancy methods that meet the standards of the traditional button system.
- Simplicity
    - Must be easy for clients to activate
    - Control Panel must be simple so sufficient help is provided quickly
- Safety
    - Must be safe to be kept and used in the house
    - Not a cause of harm
    - Must not affect the way a person normally lives their life

The process the robot undergoes is shown in Figure 2.

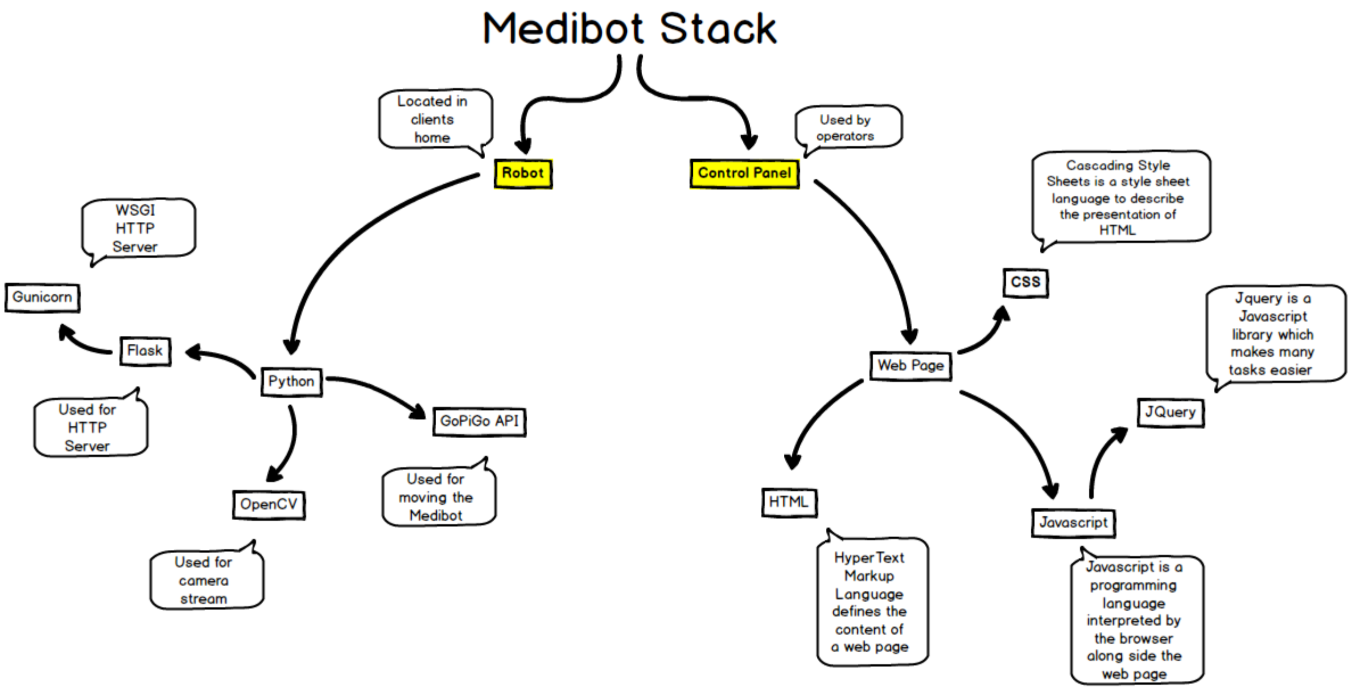
Figure 2 Initial Plan



Technical Information

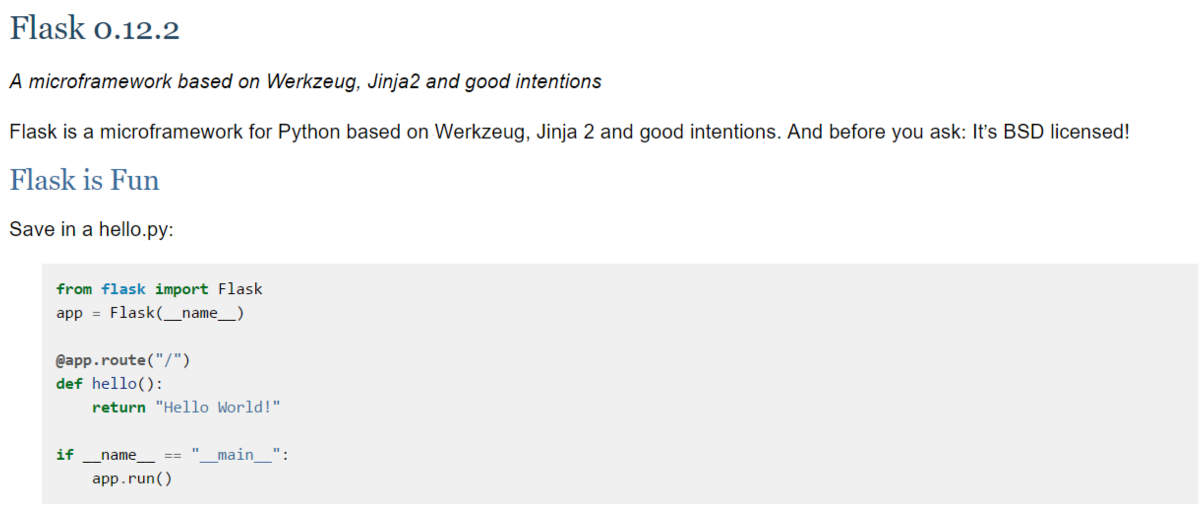
Figure 3 shows the stack of our Medibot. Our stack is the pieces of software and the tools we used to achieve our aims.

Figure 3 Diagram showing Medibot stack


Medibot Stack
Located in clients home
Robot
Control Panel
Used by operators
WSGI HTTP Server
Gunicorn
Flask
Used for HTTP Server
Python
GoPiGo API
Used for moving the Medibot
OpenCV
Used for camera stream
Cascading Style Sheets is a style sheet language to describe the presentation of HTML
CSS
Jquery is a Javascript library which makes many tasks easier
Web Page
JQuery
HTML
Javascript
HyperText Markup Language defines the content of a web page
Javascript is a programming language interpreted by the browser along side the web page

- Robot
  - HTTP Server
    - Executing commands on the robot is done by hosting a HTTP server.
    - This works by responding to pages with certain commands
    - E.g. when the page “/move?drcn=forward” is accessed the robot will begin to move forward
    - We used Flask – a Python module, to make serving the requests simple
    - The 7 lines of code in the documentation for Flask (seen in Figure 4 Screenshot from Flask documentation) is a “Hello World!” example for the module.

Figure 4 Screenshot from Flask documentation



Flask 0.12.2

A microframework based on Werkzeug, Jinja2 and good intentions

Flask is a microframework for Python based on Werkzeug, Jinja 2 and good intentions. And before you ask: It's BSD licensed!

Flask is Fun

Save in a hello.py:

```python
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```

- GoPiGo API
  - The movement of the robot is controlled with the GoPiGo API
  - It is a module developed by the manufacturer of the circuit board we use
  - Allows handling of the servo and motors
- Camera Stream
  - Used the OpenCV module
  - Frames which the stream is composed of are sent though Flask
  - Initially used MJPEG streamer but we found that there was too much latency

- Control Panel.
  - Composed of:
    - HTML
      - Is a mark-up language that defines the content of the page
    - CSS
      - Stands for Cascading Style Sheets and stylises the HTML content
      - Foundation is a CSS framework we used to help us design the layout of the control panel
    - JS
      - Is a programming language interpreted by the browser to add interactivity to the website
      - JQuery is a Javascript library that was used to detect key presses to move the robot
  - A screenshot of the control panel can be seen at Figure 5 Screenshot of control panel
  - The control panel works by sending HTTP requests to the Medibot
  - The camera stream hosted by the Medibot is also visible on the control panel

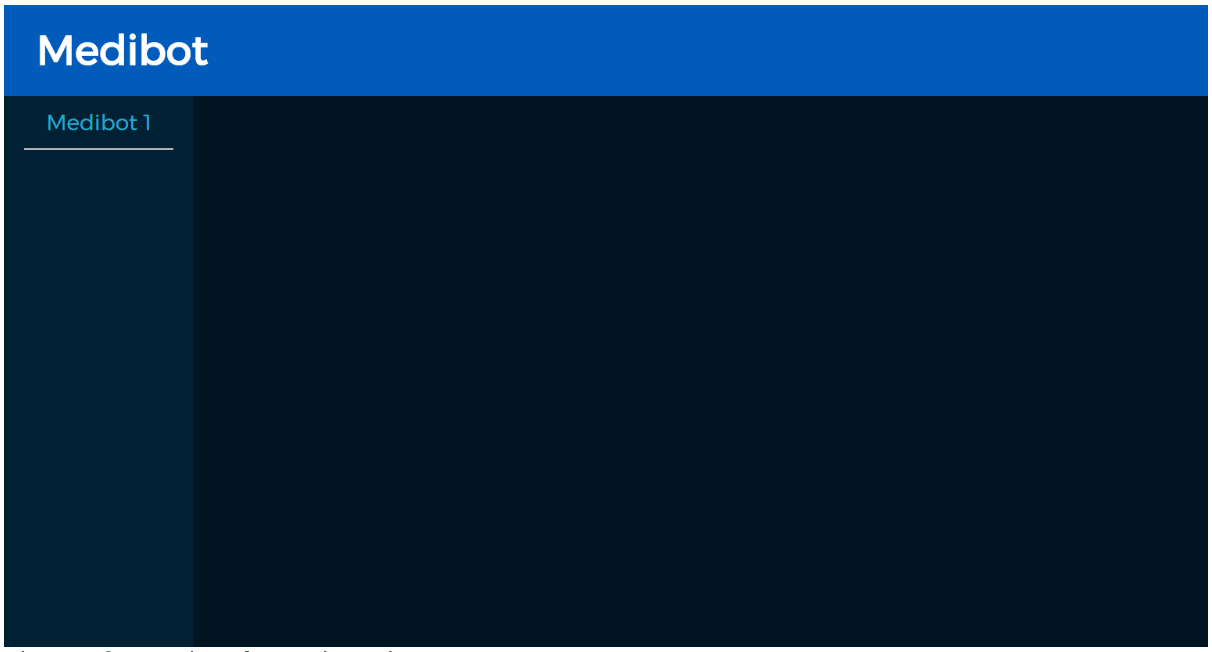


Figure 5 Screenshot of control panel

Robot specification

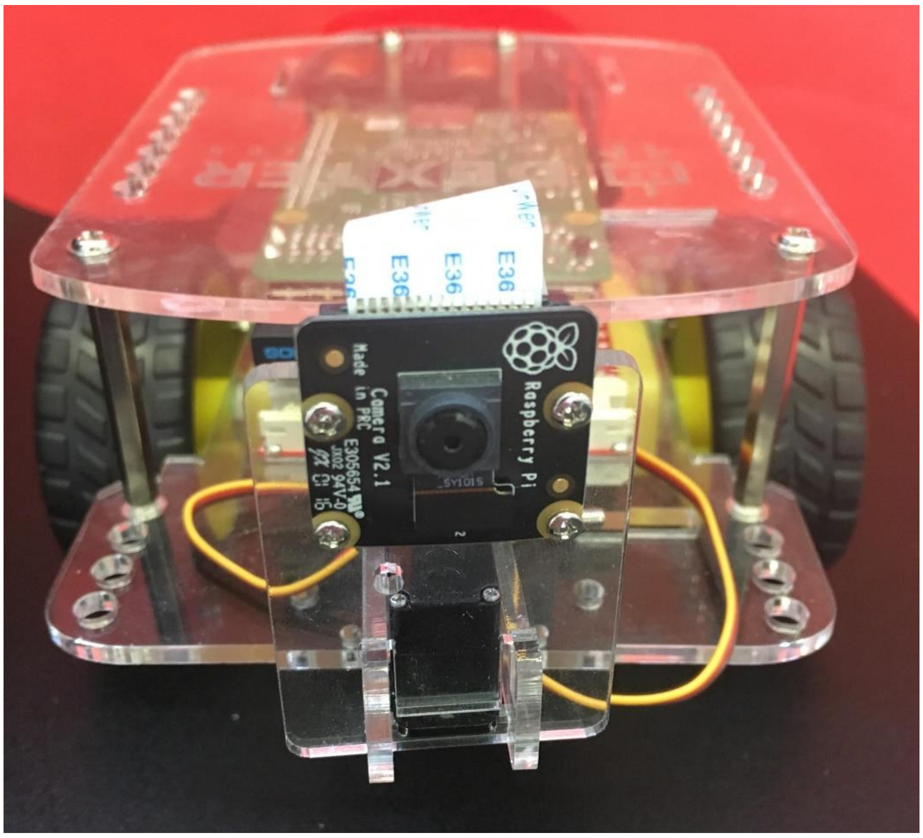
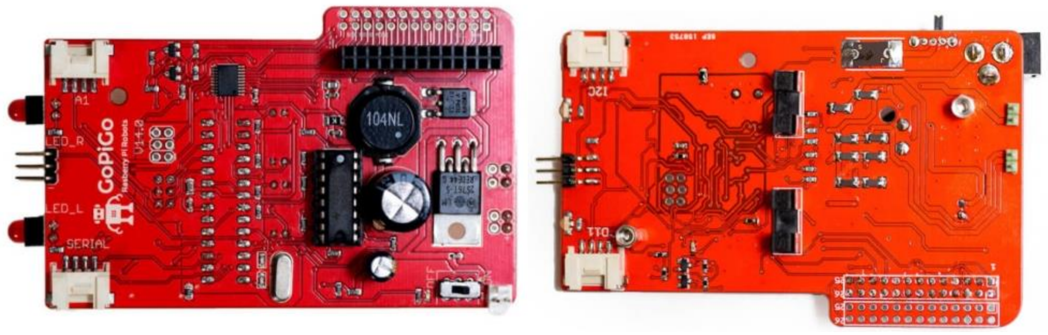


Figure 6 The robot we constructed composed of the parts described below

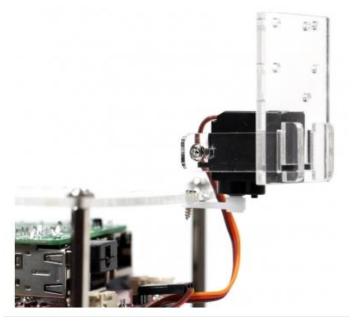
To make the Medibot, we used a variety of devices and instruments which all serve a purpose as a part within our robot. Images of these devices are shown below with a brief explanation as to what each part is and what it does.

- **GoPiGo circuit board and motor controller** - this controls the servo and motors though the Raspberry Pi that this board is fastened to.



- **Raspberry Pi 2 B** – A microcomputer which stores and executes the code, it also uses the GoPiGo circuit board to control the servo and motors. All peripherals below excluding the servo are connected to it.

- **Camera Servo** – This servo controls the rotation of the camera which is fastened onto the chassis. It is connected to the GoPiGo instead of the Raspberry Pi.



- **Raspberry Pi NOiR camera** – This camera can provide an image when in the dark, this is central in responding to problems at night time.

- **Wi-Fi dongle** – This USB device connects the robot wirelessly to a Wi-Fi network and allowed us to use VNC and SSH. VNC stands for virtual network computing and which uses remote desktop into our robot. Likewise, SSH allows for remote access to our robot in a terminal.

802.11n

Evaluation

- Achieved its aims of allowing visual communication
- We would need to continue development so that it could be deployed and replace the current alarm system.

- We used Python
  - Because of previous experience
  - Most effective because it already had an array of modules we could use
- We could use sockets
  - The use of low level sockets would increase the speed and reliability of communication
- Our software stack changed throughout development
  - We were previously using the .NET Framework for the control panel
  - We concluded that a web based control panel would be more effective because it could be interpreted on multiple platforms

- Going further:
  - Docking Station
    - We would like to develop a docking station to charge the robot and provide audio communication.
    - This would lower maintenance because of the automatic charging.
  - Outer casing
    - To improve the safety of the robot we would like to develop an outer casing,
    - This would protect the person and the robot
    - We would make it brightly coloured for any partially blind people
      - www.visionaware.org states that "Solid, bright colours, such as red, orange, and yellow are usually more visible than pastels." so we would therefore aim to create a red casing.

- We think that our project has a real-world application as it has the ability to help thousands of vulnerable people live more independently.