Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 7 – No. 8, October 2017 – www.caeaccess.org

An Introduction to Hybrid Control Architecture based on Multi-agent System for a Hexacopter Autonomy Optimization

Redouane Dargham
Team Architecture of Systems
ENSEM, BP 8118, Oasis, Casablanca, Morocco

Hicham Medromi
Team Architecture of Systems
ENSEM, BP 8118, Oasis, Casablanca, Morocco

Adil Sayouti
Team Architecture of Systems
ENSEM, BP 8118, Oasis, Casablanca, Morocco

ABSTRACT

An unmanned aerial vehicle UAV (also known as a drone) refers to a pilotless aircraft. Next generation of this kind of flying robots will be designed to fly autonomously at long range and high endurance for multi-complex purposes as surveillance, reconnaissance, electronic warfare and target acquisition. For operating in uncontrolled and dynamic environments, the robot must constantly reconfigure itself to adapt to the external conditions and its own goals. The need of control architecture to manage these reconfigurations becomes the first order of business to improve the autonomy of such robots. Such control is achieved through the use of computer systems. But "truly" autonomous UAV is done by placing a computer onboard the UAV with efficient control architecture, allowing it to process all commands itself. The most popular system meeting these criteria is the Raspberry Pi and Arduino families of microcontrollers. In the other hand an array of sensors including a GPS, an IMU and a camera are used. Through using of software state of the art and modern telecommunication networks, this aerial robots will be capable of intelligent navigation (multi-agent control architecture) combining GPS (Global Positioning System) in outdoor environment through the use of a machine interface employing mapping software maintaining an internet connection through 4G LTE and image processing in indoor space utilizing various image processing algorithms. An illustration of this study will be given in an application of control of an autonomous hexacopter developed by the team architecture of systems, in the national engineering school of electricity and mechanic in Casablanca in Morocco.

Keywords

Vertical Take Off And Landing Unmanned Aerial Vehicle (VTOL UAV), autonomy, multi-agents system, control architecture, Raspberry Pi ,4G LTE, image processing and Hexacopter

1. INTRODUCTION

Unmanned aerial vehicles (UAV) refers to a pilotless aircraft, are useful for many applications where human intervention is considered difficult or dangerous. Traditionally, the fixed-wing UAV has been served as the unit for these dangerous tasks because the control is easy. Rotorcraft UAV (RUAV), on the other hand, can operate in many different flight modes which the fixed-wing one is unable to achieve, such as vertical take-off/landing, hovering, lateral flight and pirouette. These characteristics make RUAV applicable for many military and civil applications particularly surveillance, reconnaissance, electronic warfare and target acquisition. These robots are flying devices that can autonomously be programmed or remotely controlled, either by a remote control or a ground station. A radio-controllable (RC) miniature aircrafts have been around for a long time now. The original radio control scheme, using Pulse Width Modulation (PWM) over Very High Frequency (VHF) radio bands, is interference sensitive. Nowadays, the radio technologies have evolved and the modern schemes tend to use less interference sensitive proprietary schemes, such as the 2.4 GHz Frequency Hopping Spread Spectrum (FHSS) used in the Futaba controllers (Futaba, 2015). A common factor for both schemes is the relatively restricted both range and autonomy. The development of electronics has changed the simple RC aircraft into an intelligent automated drone, with Global Positioning System (GPS) enabling up-to date location information. Furthermore, the advent of radio and video camera technologies has made live video streams possible in drones which facilitate breaking the range restriction of the old radio schemes. "Smart drone" is a more modern term, inferring that sensors within these RUAVs feed into a network infrastructure where drones are connected to other devices via Internet technologies, which enables communication. One option is the 4th generation cellular network standard, Long Term Evolution (LTE) which has a low latency and high bit rate. The main goal of this paper is to increase the autonomy of this kind of aerial robots. At this point, the first goal is to build hardware and software architecture based on controlling the multicopter over LTE, the second goal is to use the vision computer for indoor localization and the third goal is to stream a live video from the drone. In this optic the outline of this paper is as follows: Part 2 concerns a physical architecture, in part 3 multi-agent architecture is shown, in part 4 internet of thing integration and proposed solutions are depicted.

2. PHYSICAL ARCHITECTURE

2.1 Hexacopter Description

By Gentile (2012) a multicopter is an aircraft which uses multiple upwards directed rotors and which is controlled by variating the rotors speeds. All of this is controlled by powerful micro-electronics and sensors Fig1. In a hexacopter configuration, the six rotors must each rotate counter to the rotors directly adjacent to them in order to counteract the torque experienced by the UAV, so as to generate lift.

Rotation of the hexacopter is achieved by speeding up individual motors relative to the others, causing an imbalance resulting in a yaw rotation. Vertical thrust is generated by spinning all rotors faster, and horizontal travel is generated by decreasing the thrust of rotors on a certain side of the hexacopter, causing it to move in that direction. When compared to a helicopter (two rotors) or quadcopter (four rotors), the extra rotors of a hexacopter offer several additional capabilities. A hexacopter features greater lifting strength and stability, along with the ability to lose one motor and still have sufficient power to control the UAV.

Fig.1 ENSEM hexacopter

2.2 Architecture Design

2.2.1 Microcontrollers

UAVs can be controlled directly by an operator or they can be controlled by a pre-programmed computer. UAVs controlled by a computer with the ability to complete tasks with no human interaction are known as autonomous UAVs. The computer can be located on the ground, from where it can send commands wirelessly which will be processed and carried out by the UAV. Alternatively, a computer can be installed on the UAV, allowing the UAV to act completely independently. Such a computer system must be compact and lightweight; the most popular systems meeting these criteria are the Arduino family of microcontrollers and the Raspberry Pi.

Arduino The Arduino [4] is a popular single-board microcontroller designed around an 8-bit Atmel AVR microcontroller, or a 32-bit Atmel ARM. The Arduino is designed to be used to solve specific application problems; it is not intended to run an operating system. (Figure 2)

Figure 2 Arduino Uno

Raspberry Pi The Raspberry Pi (shown in Figure 3) includes a 700MHz ARM processor [5], a GPU, and 512MB of RAM, significantly outperforming the Arduino. Instead of executing code directly, the Raspberry Pi can run variants of the Linux operating system, allowing a greater potential of tasks to be completed. Due to its increased power over the Arduino, the Raspberry Pi can more easily handle computationally intensive tasks such as image processing, wireless communication and interfacing.

Figure 3: The Raspberry Pi 2 Model B is a credit card sized computer capable of running the Linux operating system

2.2.2Hardware design

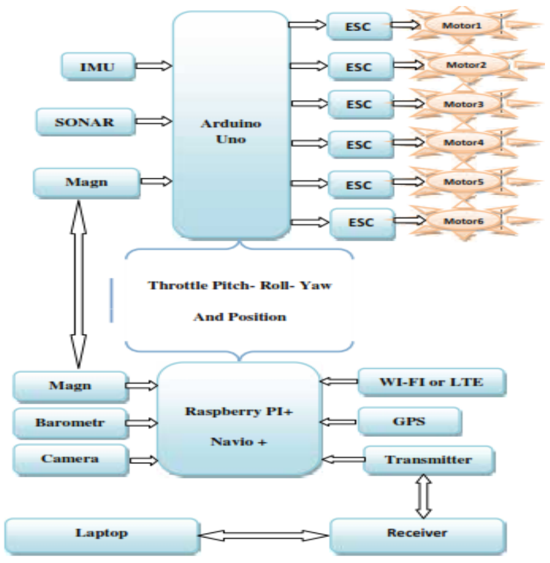


Figure 4 Hardware design

The Arduino Uno runs the main controller for balancing. It mainly takes input from the IMU and the high level controller. It continuously runs in a loop and does adjustments to individual motor to maintain a stable flight. It reads accelerometer and gyroscope values from the IMU. Once the current position and set points are calculated, a PID (proportional–integral–derivative controller) algorithm is used to calculate the individual motor values. Individual PID parameters are set to achieve a stable flight. This loop runs continuously at 300 Hz.In the other hand it can use the sonar device for collision avoidance.

The Raspberry Pi, shown in Figure 3, was selected as the primary autonomous computation platform. It takes in input from the GPS, Wi-Fi module or LTE, Magnetometer, Barometer, Cameras and instructs the lower level controller to move accordingly. All related communication and image processing are done in this level. This loop may run at a lower speed than the core controller.

2.2.3 Sensors

Motor: Motors are a bit similar to normal DC motors in the way that coils and magnets are used to drive the shaft. Though the motors do not have a brush on the shaft which takes care of switching the power direction in the coils, and so it is called as brushless motors.

ESC: Electronic Speed Controller. Unlike normal motors, brushless motor has 3 phases thus it require special controller. It is capable of generating high frequency signals with controllable phases to control the speed of the motors precisely, as well as provide enough power for the motors.

Inertial Measurement Unit (IMU): which has 6DOF (6 Degrees of Freedom, 3axis Accelerometer, 3-axis Gyroscope) is used to continuously monitor the Multicopter's positions and angles. These angles are used to balance in air, and achieve a stable flight.

Magnetometer and Barometer: These sensors give the compass alignment and altitude reading of the multicopter.

GPS Module: This module gives a near to accurate latitude and longitude of the Multicopter's position.

Camera: This is used for video recording, video surveillance, and image processing for visual guidance and automated landing capabilities.

RC Transmitter and Receiver: This telemetry used to manually control the hexacopter when required.

Wi-Fi or LTE Module: This chip alone plays an important role in making the multicopter an IoT device.

Battery: The power source for the whole device.

3. SOFTWARE ARCHITECTURE

3.1 Mutiagent System

In this paper, the definition from [1] is used: An agent is a computer system that is situated in some environment, and is capable of autonomous action in this environment in order to meet its design objectives. The environment can be virtual or physical (Figure 5).

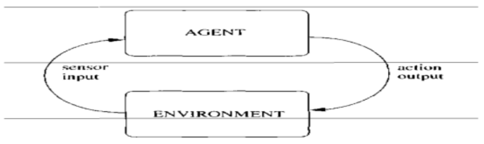


Figure 5 An agent in its environment

A multi-agent system (M.A.S.) is a computerized system composed of multiple interacting intelligent agents within an environment. The agents in a multi-agent system have several important characteristics [2]:Autonomy (the agents are at least partially independent, self-aware, autonomous) local views (no agent has a full global view of the system, or the system is too complex for an agent to make practical use of such knowledge) decentralization(there is no designated controlling agent)

3.2 Proposed Architecture

The proposed architecture is a hybrid control architecture inspired from Architecture for Autonomous system [3,6] which combines aspects of classic control and behavior-based control. This hybrid control architecture encompasses autonomy, intelligence, modularity, encapsulation, stability and parallel execution. It uses a multi-agents hybrid formalism that fits naturally the needs. The Multi-Agent System paradigm is one of the most promising approaches to create autonomous, open and dynamic systems, where heterogeneous entities are naturally represented as interacting autonomous agents, which can enter or leave the system at will. Interaction among autonomous agents is fundamental to the dynamic of multi-agent systems. Agents need to interact and coordinate their activity to carry out their common global goal. The communication between agents is realized by messages. Object oriented language is therefore absolutely suited for programming agents (C++ is chosen). Threads are used to obtain parallelism (each agent is represented by a thread in the overall process).Our architecture consists of five agents: Interface Agent, Global Planner Agent, Perception/Action Agent, Control Information Manager Agent and Communication Agent(Figure 6).

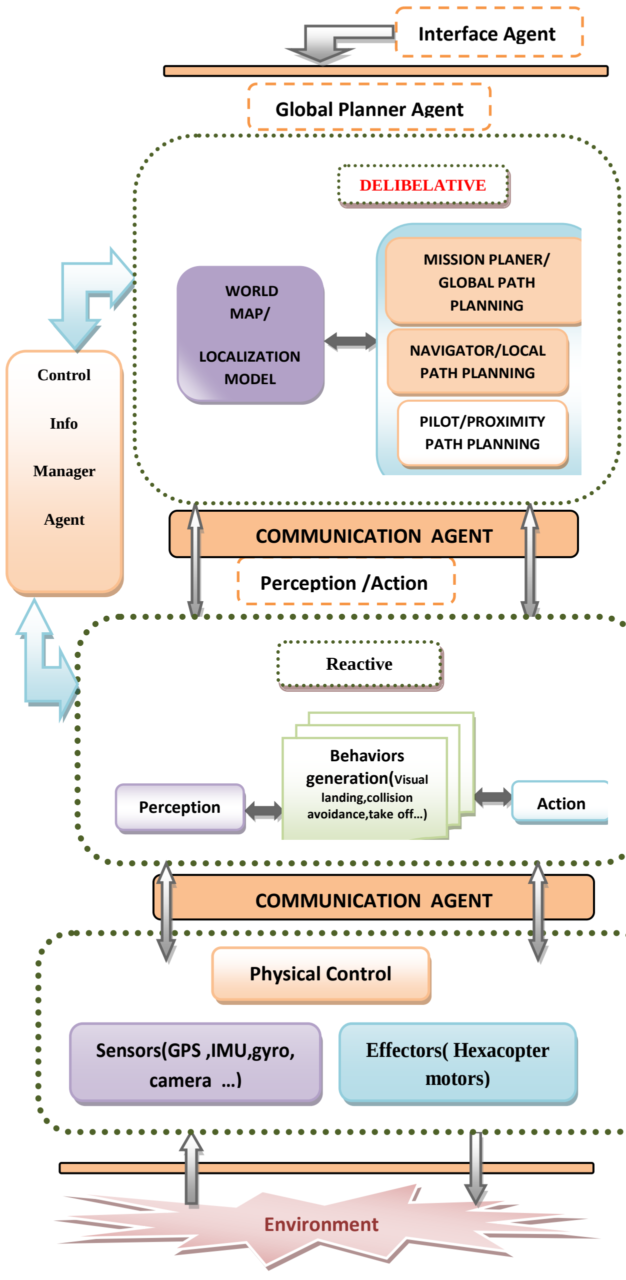

Interface Agent
Global Planner Agent
DELIBELATIVE
WORLD MAP/ LOCALIZATION MODEL
MISSION PLANER/ GLOBAL PATH PLANNING
NAVIGATOR/LOCAL PATH PLANNING
PILOT/PROXIMITY PATH PLANNING
Control Info Manager Agent
COMMUNICATION AGENT
Perception /Action
Reactive
Behaviors generation(Visual landing,collision avoidance,take off...)
Perception
Action
COMMUNICATION AGENT
Physical Control
Sensors(GPS ,IMU,gyro, camera ...)
Effectors(Hexacopter motors)
Environment

Figure 6.Proposed Architerture

-The Interface Agent is the high level of our control architecture. It must generate a succession of goal, or missions for the Global Planner Agent, according to the general mission of the robot.

-The Global Planner Agent is the foundation of the "ultimate" robot autonomy concept.It contains a mission planner, a navigator and a pilot. The Mission Planner either receives a mission from a human or generates a mission for itself .The Mission Planner can access a map in the global world model and locates where the flying robot is and where the goal is. The Navigator takes this information and generates a path from the current location to the goal. It generates a set of waypoints, or straight lines for the robot to follow. The path is passed to the Pilot. The Pilot takes the first straight line or path segment and determines what actions the robot has to do to follow the path segment. The function of the pilot is to convert this trajectory into orders to be performed by the perception/action agent.all computing are done in the Raspberry PI.

-The perception/action agent is composed of perception and action modules (Arduino Uno) .It manages the processing of incoming data (the sensors measurements) for maintaining a stable flight and creates representations of the environment, then chooses the robot behavior (Visual landing, collision avoidance, take off…) according to all information available and necessary to this choice: the fixed goal, representations and the robot localization.

-The communication agent is an interface between the software architecture and real robot. Changing the real robot requires the use of a specific agent but no change in the overall architecture.

-Control Information Manager Agent is responsible for controlling the status of each agent, ensures the exchange between the high level controller and low level controller.

3.3 The Software Controller

In the absence of knowledge of the underlying process, a PID controller has historically been considered to be the most useful controller. By tuning the three parameters in the PID controller algorithm, the controller can provide control action designed for specific process requirements. The response of the controller can be described in terms of the responsiveness of the controller to an error, the degree to which the controller overshoot the set point, and the degree of system oscillation.

In the following some results from Matlab show how tuning a control loop by adjusting the PID control parameters (proportional band/gain, integral gain/reset, derivative gain/rate) help to reach the optimum values for the desired control response[7].For the proportional term $P=K_p e(t)$ some results are given for $K_i=K_d=0$

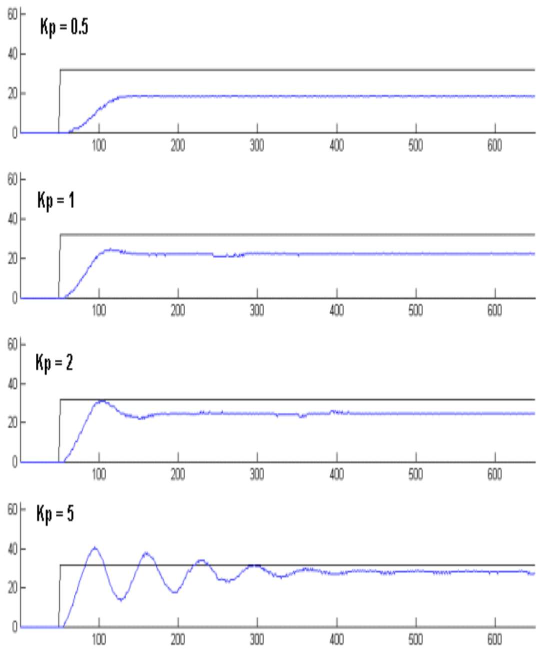


Figure.7.The evolution of offset the Kp values

With a proportional controller offset (deviation from set-point) is present (Figure.7). Increasing the controller gain will make the loop go unstable. Integral action was included in controllers to eliminate this offset.

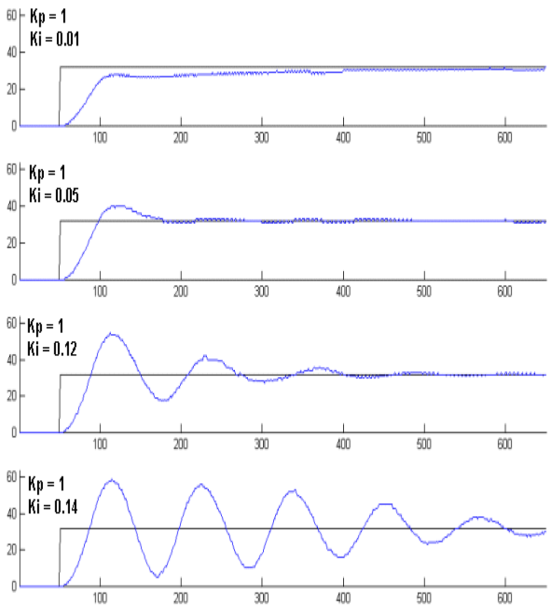


Figure.8.The offset evolution with the Ki values

Integral action has eliminated the offset Figure.8. The response is somewhat oscillatory and can be stabilized some by adding derivative action.

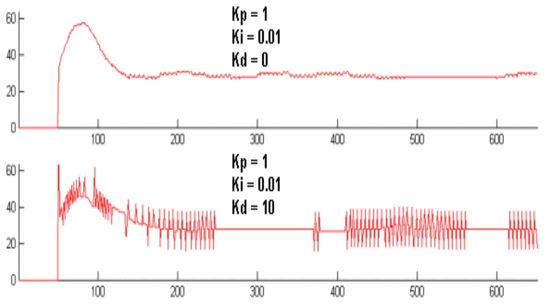


Figure.9.The offset evolution with the Kd values

Derivative action can compensate for a changing measurement (Figure.9). Thus derivative takes action to inhibit more rapid changes of the measurement than proportional action. When a load or set-point change occurs, the derivative action causes the controller gain to move the "wrong" way when the measurement gets near the set-point. Derivative is often used to avoid overshoot.

Derivative action can stabilize loops since it adds phase lead. Generally, the derivative action is used, more controller gain and reset can be used.

As the hexacopter is an under-actuated aerial robot, the architecture for the control system will use two cascade PID controllers to drive it to a desired state (position, orientation, linear and angular velocity) in state space. Figure 10.

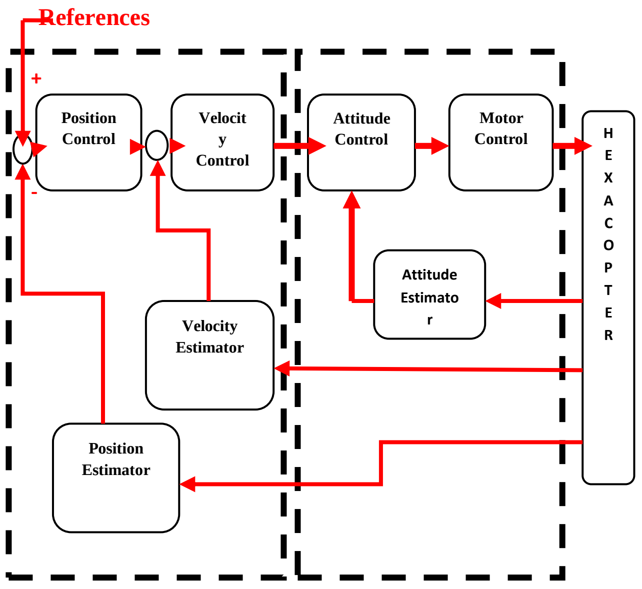


Figure.10.Figure of control system architecture

Stability (no unbounded oscillation) is a basic requirement, but beyond that, different systems have different behavior, different applications have different requirements, and requirements may conflict with one another. Usually, initial designs need to be adjusted repeatedly through computer simulations until the closed-loop system performs or compromises as desired.

Generally, stabilization of response is required and the process must not oscillate for any combination of process conditions and set points, though sometimes marginal stability (bounded oscillation) is acceptable or desired.

4. INTERNET OF THING INTEGRATION

Internet of Things (IOT) is loosely-coupled, decentralized network of autonomous, yet cooperating, intelligent heterogeneous devices. The IoT devices are capable of sensing, actuating, storing data, data and information interpretation and data exchange, in situated context-aware manner [7], [8], [9], [10].Challenges in the IOT include addressing device heterogeneity, interoperability issues with different communication technologies, cooperation coordination and scalability beyond current systems.
Recently, agent-based systems have been suggested for IOT to provide autonomy, proactive and reactive features, anthologies for cooperation and different contexts. Software agents in IOT devices are proposed to coordinate cooperation, allow mobility of service components, encapsulate the device functionality, hide specific communication details and abstract the heterogeneous device hardware. Software agents in the IOT devices are required to provide computational capabilities and interfaces for communication.

4.1 The System Architecture

In the IOT devices, the software agents provide atomic system services, such as communication interfaces and runtime execution of the task code. The task is executed according to the instructions in the received message. First, before the execution, the agent needs to query the utilized resource states for the given resource URLs, then it needs to map the states into the task code, after which it can execute the task code. Finally, it needs to update the state based on the task execution results. The agent is aware of only these instructions during the task execution, which allows the resources in the device to be utilized in different tasks, contexts or applications without considerable overhead [11], [12].In the hardware architecture presented in this paper the IOT configuration is presented in Figure.11

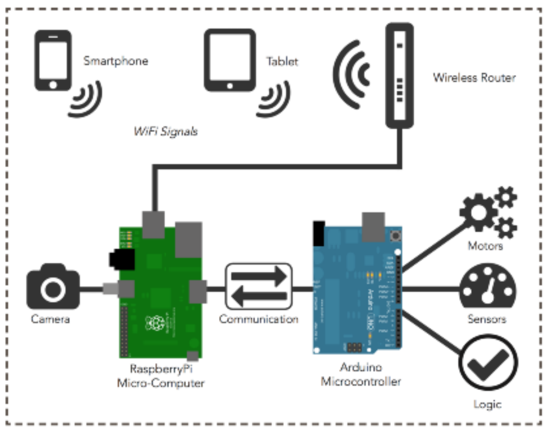
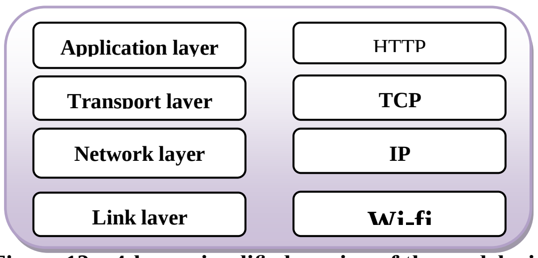


Figure.11.Figure of IOT configuration with chosen hardware

4.2 Wireless Communication

An autonomous UAV must be supplied directives to complete tasks; this can either occur during construction or operation. In order to send the UAV directives, a wireless communication protocol must be chosen. There are different types of wireless technologies relevant for IOT; these technologies span different spaces from few centimeters to many kilometers. For short to medium range communication Wireless Personal and Local Area Network technologies (WPAN\LAN) such as: Bluetooth, ZigBee, 6LowPAN, and Wi-Fi are recommended. For long range communication the recommendation is for Wireless Wide Area Network technologies (WWAN) the well known technology is cellular 2G/3G/4G and 5G in future.

Communication systems utilize a set of rules and standards to format data and control data exchange. The most common model in data communication systems is the Open Systems Interconnection (OSI) model, which breaks the communication into functional layers allowing easier implementation of scalable and interoperable networks. The OSI model has 7 layers; a 4-layer simplified version of the model is shown in Figure 12 along with example of the TCP/IP stack.



Application layer	HTTP
Transport layer	TCP
Network layer	IP
Link layer	Wi-fi

Figure.12 a 4-layer simplified version of the model with example of the TCP/IP stack

The **Link layer** provides conversion of bits to radio signals (and vice versa), takes care of data framing for reliable wireless communication and manages the access to the radio channel. In the TCP/IP example in Figure 2, Wi-Fi is shown as the link layer protocol.

The **Network layer** addresses and routes data through the network. IP (Internet Protocol) is the network layer protocol of the Internet, providing an IP address to devices and carrying IP packets from one device to another.

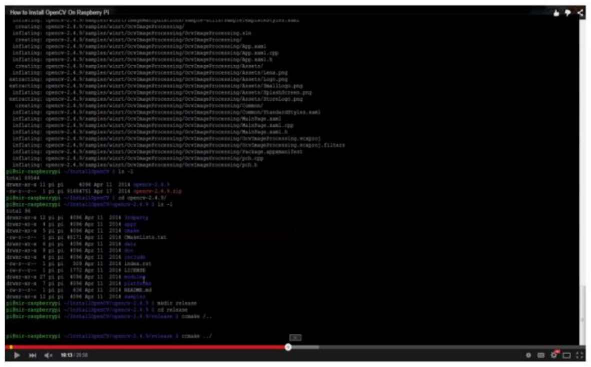
The **Transport layer** generates communication sessions between applications running on two ends of the network. It allows multiple applications to run on one device, each using its own communication channel. TCP (or Transmission Control Protocol) is the predominant transport protocol in the Internet.

The **Application layer** is responsible for data formatting and it governs the data flow in an optimal scheme for specific applications. A popular application layer protocol in the TCP/IP stack is HTTP (or Hypertext Transfer Protocol) which was created to transfer web content over the Internet.

LTE-Advanced technology, the chief vehicle of 4G cellular connectivity, started to and will continue evolving to provide new features that support a range of high and low performance and cost-optimized IoT device categories. Such devices also support extended coverage for challenging locations, low energy consumption for applications requiring long battery life and optimizations to support very large numbers of devices per cell.

4.3 Image Processing

The process describes the method used for an object detection.The goal will be: to detect objects of the color orange.OpenCV contains libraries and software written to perform algorithms that would otherwise be extremely time consuming to calculate for each and every function to be called. It helps the program not just see, but understand the world. In a sense, it could be considered a branch of remote sensing technology. OpenCV gaves the benefit of not having to learn about bit depth , buffer management, Eigen values, among other complicated computer vision parameters. The only downside to this was installing OpenCV on the RPi. [13].(Figure.13)



```
  creating: opencv-2.4.9/samples/winrt/OcvImageProcessing/
 inflating: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing.sln
  creating: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing/
 inflating: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing/App.xaml
 inflating: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing/App.xaml.cpp
 inflating: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing/App.xaml.h
  creating: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing/Assets/
 inflating: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing/Assets/Lena.png
extracting: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing/Assets/Logo.png
extracting: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing/Assets/SmallLogo.png
 inflating: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing/Assets/SplashScreen.png
extracting: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing/Assets/StoreLogo.png
  creating: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing/Common/
 inflating: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing/Common/StandardStyles.xaml
 inflating: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing/MainPage.xaml
 inflating: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing/MainPage.xaml.cpp
 inflating: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing/MainPage.xaml.h
 inflating: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing/OcvImageProcessing.vcxproj
 inflating: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing/OcvImageProcessing.vcxproj.filters
 inflating: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing/Package.appxmanifest
 inflating: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing/pch.cpp
 inflating: opencv-2.4.9/samples/winrt/OcvImageProcessing/OcvImageProcessing/pch.h
pi@[illegible]-raspberrypi ~/InstallOpenCV $ ls -l
total [illegible]
drwxr-xr-x 11 pi pi     4096 Apr 11  2014 opencv-2.4.9
-rw-r--r--  1 pi pi [illegible] Apr 17  2014 opencv-2.4.9.zip
pi@[illegible]-raspberrypi ~/InstallOpenCV $ cd opencv-2.4.9/
pi@[illegible]-raspberrypi ~/InstallOpenCV/opencv-2.4.9 $ ls -l
total 96
drwxr-xr-x 12 pi pi  4096 Apr 11  2014 3rdparty
drwxr-xr-x  4 pi pi  4096 Apr 11  2014 apps
drwxr-xr-x  5 pi pi  4096 Apr 11  2014 cmake
-rw-r--r--  1 pi pi 48171 Apr 11  2014 CMakeLists.txt
drwxr-xr-x  6 pi pi  4096 Apr 11  2014 data
drwxr-xr-x  9 pi pi  4096 Apr 11  2014 doc
drwxr-xr-x  4 pi pi  4096 Apr 11  2014 include
-rw-r--r--  1 pi pi   309 Apr 11  2014 index.rst
-rw-r--r--  1 pi pi  1772 Apr 11  2014 LICENSE
drwxr-xr-x 27 pi pi  4096 Apr 11  2014 modules
drwxr-xr-x  7 pi pi  4096 Apr 11  2014 platforms
-rw-r--r--  1 pi pi   436 Apr 11  2014 README.md
drwxr-xr-x 12 pi pi  4096 Apr 11  2014 samples
pi@[illegible]-raspberrypi ~/InstallOpenCV/opencv-2.4.9 $ mkdir release
pi@[illegible]-raspberrypi ~/InstallOpenCV/opencv-2.4.9 $ cd release
pi@[illegible]-raspberrypi ~/InstallOpenCV/opencv-2.4.9/release $ cmake /..

pi@[illegible]-raspberrypi ~/InstallOpenCV/opencv-2.4.9/release $ cmake ../
```

Figure 13: Installing OpenCV on RPi

5. CONCLUSION AND FUTURE WORK

The expected benefits of this hybrid multi-agent architecture based IOT include: minimal task-based set of system resources are utilized in each computation, there is no need for task-based system configuration, locality can be exploited in the execution of task and service dynamically based on the instructions, context-switching in the devices is a matter of the instructions in the messages and the computational load is distributed among the participating devices. Additionally, mobility of the system devices is supported and this method allows both participatory and opportunistic application scenarios.

Our future work includes the first real-world demonstrations of this architecture and to develop an interface utilizing existing web technologies for a uniform experience across a multitude of devices and browsers. The blueMix platform will be used and the hexacopter may be connected through a Wi-Fi network, or through 4G LTE, allowing for control at great distances.

6. REFERENCES

[1] Wooldridge, N. R. Jennings et al., “Intelligent agents: Theory and practice,” Knowledge engineering review, vol. 10, no. 2, pp. 115–152,1995. .

[2] M. Wooldridge, An introduction to multiagent systems. Wiley, 2008.

[3] Adil Sayouti, Hicham Medromi, F. Qrichi Aniba Control Architecture design for a Mobile Robot via the Internet 9th International PhD Workshop on system and control ,October 2008,Slovinia.

[4] W. Durfee, University of Minnesota oct-2011 Available on-line at www.me.umn.edu/courses/me2011/arduino/

[5] Raspberry Pi® User Guide Eben Upton and Gareth Halfacree 2012 John Wiley & Sons Ltd.

[6] Wooldridge,Michael (2012). *An Introduction to MultiAgent Systems*. John Wiley & Sons. p. 366. ISBN 0-471-49691-X.

[7] A.Visioli,“Practical PID Control”, Springer-Verlag, London 2006.

[8] G. Fortino, A. Guerrieri, and W. Russo, "Agent-oriented smart objects development," in: *16th IEEE International Conference on Computer Supported Cooperative Work in Design*, pp. 907-912, IEEE, 23-25 May, 2012

[9] I. Ayala, M. Amor, and L. Fuentes, "An agent platform for self-configuring agents in the Internet of Things," in: *The 3rd International Workshop on Infrastructures and Tools for Multiagent Systems*, June 5, Valencia, Spain, 2012.

[10] E. Kazanavicius, and L. Ostaseviciute, “Agent-based framework for embedded systems development in smart environments,” in: *15th International Conference on Information and Software Technologies*, p. 194-200, Kaunas, Lithuania, April 23-24, 2009.

[11] T. Leppänen, P. Närhi, J. Ylioja, J. Riekki, Y. Tobe and T.Ojala, “On using continuations in wireless sensor networks,”in: *9th International Conference on Networked.*