Benha University

Faculty of Engineering

Computer Department

**OBJECT**

**RECOGNITION**A Computer Vision Based

Intelligent Mobile Robot

Supervised by:

Prof. Abeer Twakol



**Benha University  
Faculty of Engineering  
Computer Department**

Presented to: **Prof. Abeer Twakol**

Presented by:

Ibrahim Ahmed Taher  
Ahmed Rafik Mohammed   
Ahmed Lotfy Abdel Fattah  
Ahmed Safwat Abdel Aziz   
Ameen Mohammed Ameen  
Kariem Elsayed Mostafa

**Acknowledgment**

Any attempt at any level cannot be satisfactorily completed without the will of God and the Support and guidance of learned people. We would like to express our immense gratitude to *Prof. Abeer Twakol* and *Eng. Abdelrahman Elewah* for their constant support and motivation that have encouraged us to come up with this project.

**Abstract**

Today’s needs and efforts towards advancing technology for humanity is limitless. The way Machine Learning is changing how we perceive information, it has inspired solutions for a variety of everyday problems. With the advent of Machine Learning, objects will be capable of making their own decisions. Being an interesting area of research, there is a variety of techniques and algorithms for Machine Learning. Nonetheless we may not forget Object recognition that is certainly one of the most exciting areas in machine learning right now. Computers have been able to recognize different objects even to the finest details like gestures and facial expressions, but recognizing arbitrary objects within a larger image has been the real deal. Surprisingly our brains effortlessly convert photons bouncing off objects at slightly different frequencies into a spectacularly rich set of information about the world around us. Machine learning still struggles with these simple tasks, but in the past few years, it’s gotten much better

Table of Content

[ACHNOWLEDGMENT i](#_Toc506057936)

[ABSTRACT Ii](#_Toc506057936)

[table of content IIi](#_Toc506057937)

[Introduction](#_Toc506057939) IV

Machine Revolution, Preluded

[1.1 Where It All Started 1](#_Toc506057940)

[**What’s Machine Learning?** 1](#_Toc506057941)

[**Examples of Machine Learning Applications** 3](#_Toc506057942)

[1.2 Computer Vision 8](#_Toc506057943)

[**Vision** 8](#_Toc506057944)

[**Recognition** 9](#_Toc506057945)

[**Understanding** 11](#_Toc506057946)

[1.3 Object Recognition 12](#_Toc506057947)

[**Computer Vision Applications** 13](#_Toc506057948)

Project Alpha

[2.1 Embedded System 15](#_Toc506057949)

[**Definitions** 15](#_Toc506057950)

[**Characteristics of an Embedded System** 16](#_Toc506057951)

[**Basic Structure of an Embedded System** 17](#_Toc506057952)

[**Processors** 18](#_Toc506057953)

[**Microcontroller** 19](#_Toc506057954)

[**Arduino vs Raspberry Pi** 27](#_Toc506057955)

[2.2 The Web 31](#_Toc506057956)

[**HTTP Protocol** 31](#_Toc506057957)

[**Web Servers** 33](#_Toc506057958)

[**Databases** 36](#_Toc506057959)

[**LAMP Stack** 38](#_Toc506057960)

[2.3 The Project 42](#_Toc506057961)

# **Introduction**

The last few years have seen an explosion in the machine learning field, object recognition and artificial intelligence. The lowered cost of processors along with increasing the processing capabilities and getting introduced to new Nano-technologies has resulted in many products being made “smart” and able to make smart decisions.

Deep learning and a large public training data set called ImageNet has made an impressive amount of progress toward object recognition. A well-known framework that is used in deep learning algorithms’ implementation on a variety of architectures. It is very good at making use of GPUs, which themselves are great at implementing deep learning algorithms.

We are trying in our project to build a robot that could recognize objects. Days of research in building computer programs and doing test-driven development have turned us into a menace working on physical projects. In our real world, testing your device for bugs can set your lap on fire, or at least burn up your components.

**CHAPTER 1:**

**MACHINE REVOLUTION, PRELUDED.**

# **Where It All Started**

## **What’s Machine Learning?**

Machine learning is an application of **artificial intelligence** (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. **Machine learning focuses on the development of computer programs** that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. **The primary aim is to allow the computers learn automatically** without human intervention or assistance and adjust actions accordingly.

### **Machine learning methods**

Machine learning algorithms are often categorized as supervised or unsupervised.

* **Supervised machine learning algorithms**can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.
* In contrast, **unsupervised machine learning algorithms**are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn’t figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.
* **Semi-supervised machine learning algorithms** fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring unlabeled data generally doesn’t require additional resources.
* **Reinforcement machine learning algorithms**is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information.

## **Examples of Machine Learning Applications**

Artificial Intelligence (AI) is everywhere. Possibility is that you are using it in one way or the other and you don’t even know about it. One of the popular applications of AI is Machine Learning (ML), in which computers, software, and devices perform via cognition (very similar to human brain). Herein, we share few examples of machine learning that we use every day and perhaps have no idea that they are driven by ML.

**1. Virtual Personal Assistants**

Siri, Alexa, Google Now are some of the popular examples of virtual personal assistants. As the name suggests, they assist in finding information, when asked over voice. All you need to do is activate them and ask “What is my schedule for today?”, “What are the flights from Germany to London”, or similar questions. For answering, your personal assistant looks out for the information, recalls your related queries, or send a command to other resources (like phone apps) to collect info. You can even instruct assistants for certain tasks like “Set an alarm for 6 AM next morning”, “Remind me to visit Visa Office day after tomorrow”.

Machine learning is an important part of these personal assistants as they collect and refine the information on the basis of your previous involvement with them. Later, this set of data is utilized to render results that are tailored to your preferences.

Virtual Assistants are integrated to a variety of platforms. For example:

* Smart Speakers: Amazon Echo and Google Home
* Smartphones: Samsung Bixby on Samsung S8
* Mobile Apps: Google Allo

**2. Predictions while Commuting**

*Traffic Predictions*: We all have been using GPS navigation services. While we do that, our current locations and velocities are being saved at a central server for managing traffic. This data is then used to build a map of current traffic. While this helps in preventing the traffic and does congestion analysis, the underlying problem is that there are less number of cars that are equipped with GPS. Machine learning in such scenarios helps to estimate the regions where congestion can be found on the basis of daily experiences.

*Online Transportation Networks*: When booking a cab, the app estimates the price of the ride. When sharing these services, how do they minimize the detours? The answer is machine learning. Jeff Schneider, the engineering lead at Uber ATC reveals in an interview that they use ML to define price surge hours by predicting the rider demand. In the entire cycle of the services, ML is playing a major role.

**3. Videos Surveillance**

Imagine a single person monitoring multiple video cameras! Certainly, a difficult job to do and boring as well. This is why the idea of training computers to do this job makes sense.

The video surveillance system nowadays are powered by AI that makes it possible to detect crime before they happen. They track unusual behavior of people like standing motionless for a long time, stumbling, or napping on benches etc. The system can thus give an alert to human attendants, which can ultimately help to avoid mishaps. And when such activities are reported and counted to be true, they help to improve the surveillance services. This happens with machine learning doing its job at the backend.

**4. Social Media Services**

From personalizing your news feed to better ads targeting, social media platforms are utilizing machine learning for their own and user benefits. Here are a few examples that you must be noticing, using, and loving in your social media accounts, without realizing that these wonderful features are nothing but the applications of ML.

* *People You May Know*: Machine learning works on a simple concept: understanding with experiences. Facebook continuously notices the friends that you connect with, the profiles that you visit very often, your interests, workplace, or a group that you share with someone etc. On the basis of continuous learning, a list of Facebook users are suggested that you can become friends with.
* *Face Recognition*: You upload a picture of you with a friend and Facebook instantly recognizes that friend. Facebook checks the poses and projections in the picture, notice the unique features, and then match them with the people in your friend list. The entire process at the backend is complicated and takes care of the precision factor but seems to be a simple application of ML at the front end.
* *Similar Pins*: Machine learning is the core element of Computer Vision, which is a technique to extract useful information from images and videos. Pinterest uses computer vision to identify the objects (or pins) in the images and recommend similar pins accordingly.

**5. Email Spam and Malware Filtering**

* There are a number of spam filtering approaches that email clients use. To ascertain that these spam filters are continuously updated, they are powered by machine learning. When rule-based spam filtering is done, it fails to track the latest tricks adopted by spammers. Multi-Layer Perceptron, C 4.5 Decision Tree Induction are some of the spam filtering techniques that are powered by ML.
* Over 325, 000 malwares are detected every day and each piece of code is 90–98% similar to its previous versions. The system security programs that are powered by machine learning understand the coding pattern. Therefore, they detects new malware with 2–10% variation easily and offer protection against them.

**6. Online Customer Support**

A number of websites nowadays offer the option to chat with customer support representative while they are navigating within the site. However, not every website has a live executive to answer your queries. In most of the cases, you talk to a Chabot. These bots tend to extract information from the website and present it to the customers. Meanwhile, the chatbots advances with time. They tend to understand the user queries better and serve them with better answers, which is possible due to its machine learning algorithms.

**7. Search Engine Result Refining**

Google and other search engines use machine learning to improve the search results for you. Every time you execute a search, the algorithms at the backend keep a watch at how you respond to the results. If you open the top results and stay on the web page for long, the search engine assumes that the results it displayed were in accordance to the query. Similarly, if you reach the second or third page of the search results but do not open any of the results, the search engine estimates that the results served did not match requirement. This way, the algorithms working at the backend improve the search results.

**8. Product Recommendations**

You shopped for a product online few days back and then you keep receiving emails for shopping suggestions. If not this, then you might have noticed that the shopping website or the app recommends you some items that somehow matches with your taste. Certainly, this refines the shopping experience but did you know that it’s machine learning doing the magic for you? On the basis of your behavior with the website/app, past purchases, items liked or added to cart, brand preferences etc., the product recommendations are made.

**9. Online Fraud Detection**

Machine learning is proving its potential to make cyberspace a secure place and tracking monetary frauds online is one of its examples. For example: PayPal is using ML for protection against money laundering. The company uses a set of tools that helps them to compare millions of transactions taking place and distinguish between legitimate or illegitimate transactions taking place between the buyers and sellers.

# **Computer Vision**

Assuming a ball is thrown to you and you want to catch it...  
Actually, this is one of the most complex processes we’ve ever attempted to comprehend – let alone recreate. Inventing a machine that sees like we do is a deceptively difficult task, not just because it’s hard to make computers do it, but because we’re not entirely sure how we do it in the first place.

What actually happens is roughly this: the image of the ball passes through your eye and strikes your retina, which does some elementary analysis and sends it along to the brain, where the visual cortex more thoroughly analyzes the image. It then sends it out to the rest of the cortex, which compares it to everything it already knows, classifies the objects and dimensions, and finally decides on something to do: raise your hand and catch the ball (having predicted its path). This takes place in a tiny fraction of a second, with almost no conscious effort, and almost never fails. So recreating human vision isn’t just a hard problem, it’s a set of them, each of which relies on the other.

Well, no one ever said this would be easy. Except, perhaps, AI pioneer Marvin Minsky, who famously instructed a graduate student in 1966 to “connect a camera to a computer and have it describe what it sees.” Pity the kid: 50 years later, we’re still working on it.

## **Vision**

Reinventing the eye is the area where we’ve had the most success. Over the past few decades, we have created sensors and image processors that match and in some ways exceed the human eye’s capabilities. With larger, more optically perfect lenses and semiconductor subpixels fabricated at nanometer scales, the precision and sensitivity of modern cameras is nothing short of incredible. Cameras can also record thousands of images per second and detect distances with great precision.

Yet despite the high fidelity of their outputs, these devices are in many ways no better than a pinhole camera from the 19th century: They merely record the distribution of photons coming in a given direction. The best camera sensor ever made couldn’t recognize a ball — much less be able to catch it.

The hardware, in other words, is severely limited without the software — which, it turns out, is by far the greater problem to solve. But modern camera technology does provide a rich and flexible platform on which to work.

## **Recognition**

This isn’t the place for a complete course on visual neuroanatomy, but suffice it to say that our brains are built from the ground up with seeing in mind, so to speak. More of the brain is dedicated to vision than any other task, and that specialization goes all the way down to the cells themselves. Billions of them work together to extract patterns from the noisy, disorganized signal from the retina.

Sets of neurons excite one another if there’s contrast along a line at a certain angle, say, or rapid motion in a certain direction. Higher-level networks aggregate these patterns into meta-patterns: a circle, moving upwards. Another network chimes in: the circle is white, with red lines. Another: it is growing in size. A picture begins to emerge from these crude but complementary descriptions.

Early research into computer vision, considering these networks as being unfathomably complex, took a different approach: *“top-down” reasoning — a book looks like /this/, so watch for /this/ pattern, unless it’s on its side, in which case it looks more like /this/. A car looks like /this/ and moves like /this/.*

We can barely come up with a working definition of how our minds work, much less how to simulate it.

For a few objects in controlled situations, this worked well, but imagine trying to describe every object around you, from every angle, with variations for lighting and motion and a hundred other things. It became clear that to achieve even toddler-like levels of recognition would require impractically large sets of data.

A “bottom-up” approach mimicking what is found in the brain is more promising. A computer can apply a series of transformations to an image and discover edges, the objects they imply, perspective and movement when presented with multiple pictures, and so on. The processes involve a great deal of math and statistics, but they amount to the computer trying to match the shapes it sees with shapes it has been trained to recognize — trained on other images, the way our brains were.

What an image like this one above (from Purdue University’s [E-lab](http://e-lab.github.io/)) is showing is the computer displaying that, by its calculations, the objects highlighted look and act like other examples of that object, to a certain level of statistical certainty.

Proponents of bottom-up architecture might have said “I told you so.” Except that until recent years, the creation and operation of artificial neural networks was impractical because of the immense amount of computation they require. Advances in parallel computing have eroded those barriers, and the last few years have seen an explosion of research into and using systems that imitate — still very approximately — the ones in our brain. The process of pattern recognition has been sped up by orders of magnitude, and we’re making more progress every day.

## **Understanding**

Of course, you could build a system that recognizes every variety of apple, from every angle, in any situation, at rest or in motion, with bites taken out of it, anything — and it wouldn’t be able to recognize an orange. For that matter, it couldn’t even tell you what an apple is, whether it’s edible, how big it is or what they’re used for.

The problem is that even good hardware and software aren’t much use without an operating system.

For us, that’s the rest of our minds: short and long term memory, input from our other senses, attention and cognition, a billion lessons learned from a trillion interactions with the world, written with methods we barely understand to a network of interconnected neurons more complex than anything we’ve ever encountered.

The future of computer vision is in integrating the powerful but specific systems we’ve created with broader ones.

This is where the frontiers of computer science and more general artificial intelligence converge — and where we’re currently spinning our wheels. Between computer scientists, engineers, psychologists, neuroscientists and philosophers, we can barely come up with a working definition of how our minds work, much less how to simulate it.

That said, computer vision even in its nascent stage is still incredibly useful. It’s in our cameras, recognizing faces and smiles. It’s in self-driving cars, reading traffic signs and watching for pedestrians. There’s still a long way to go before they see like we do — if it’s even possible — but considering the scale of the task at hand, it’s amazing that they see at all.

**1.3 Object Recognition**

Object recognition is the area of artificial intelligence (AI) concerned with the abilities of robots and other AI implementations to recognize various things and entities.

Object recognition allows robots and AI programs to pick out and identify objects from inputs like video and still camera images. Methods used for object identification include 3D models, component identification, edge detection and analysis of appearances from different angles, Object recognition is at the convergence points of robotics, machine vision, neural networks and AI. Google and Microsoft are among the companies working in the area -- Google’s driverless car and Microsoft’s Kinect system both use object recognition.

Robots that understand their environments can perform more complex tasks better. Major advances of object recognition stand to revolutionize AI and robotics:

MIT has created neural networks, based on our understanding of how the brain works, that allow software to identify objects almost as quickly as primates do.

Gathered visual data from cloud robotics can allow multiple robots to learn tasks associated with object recognition faster. Robots can also reference massive databases of known objects and that knowledge can be shared among all connected robots.

Scientists at Brigham Young University have developed an object recognition algorithm that can learn to identify objects on its own. The Evolution-Constructed Features algorithm, as it’s called, can make decisions about what characteristics of an object are relevant to its identification.

Concerns about the potential of object recognition include fears that advertisers and other interested entities will use the technology to mine the increasing number of images posted online and gather from them the personal information of individuals.

## **Computer Vision Applications**

**Face detection**

Popular applications include face detection and people counting. Have you ever noticed how Facebook detects your face when you upload a photo? This is a simple application of object detection that we see in our daily life.

**People Counting**

Object detection can be also used for people counting, it is used for analyzing store performance or crowd statistics during festivals. These tend to be more difficult as people move out of the frame quickly (also because people are non-rigid objects).

**Vehicle detection**

Similarly when the object is a vehicle such as a bicycle or car, object detection with tracking can prove effective in estimating the speed of the object. The type of ship entering a port can be determined by object detection (depending on shape, size etc.). This system for detecting ships are currently in development in some European countries.

**Manufacturing Industry**

Object detection is also used in industrial processes to identify products. Say you want your machine to only detect circular objects. Hough circle detection transform can be used for detection.

**Online images**

Apart from these object detection can be used for classifying images found online. Obscene images are usually filtered out using object detection.

**Security**

In the future we might be able to use object detection to identify anomalies in a scene such as bombs or explosives (by making use of a quadcopter).

**CHAPTER 2:**

**PROJECT ALPHA**

We can broadly define an embedded system as a microcontroller-based, software-driven, reliable, real-time control system, designed to perform a specific task. It can be thought of as a computer hardware system having software embedded in it. An embedded system can be either an independent system or a part of a large system.

# **2.1 Embedded System**

## **Definitions**

### **2.1.1.1 System**

A system is an arrangement in which all its unit assemble work together according to a set of rules. It can also be defined as a way of working, organizing or doing one or many tasks according to a fixed plan. For example, a watch is a time displaying system. Its components follow a set of rules to show time. If one of its parts fails, the watch will stop working. So we can say, in a system, all its subcomponents depend on each other.

### **2.1.1.2 Embedded System**

As its name suggests, Embedded means something that is attached to another thing. An embedded system can be thought of as a computer hardware system having software embedded in it. An embedded system can be an independent system or it can be a part of a large system. An embedded system is a microcontroller or microprocessor based system which is designed to perform a specific task. For example, a fire alarm is an embedded system; it will sense only smoke.

An embedded system has three components −

* It has hardware.
* It has application software.
* It has Real Time Operating system (RTOS) that supervises the application software and provide mechanism to let the processor run a process as per scheduling by following a plan to control the latencies. RTOS defines the way the system works. It sets the rules during the execution of application program. A small scale embedded system may not have RTOS.

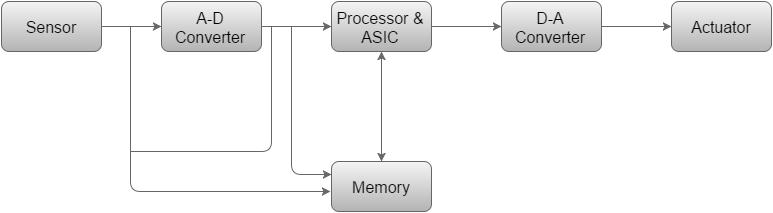
So we can define an embedded system as a Microcontroller based, software driven, reliable, real-time control system.

## **Characteristics of an Embedded System**

* **Single-functioned** − An embedded system usually performs a specialized operation and does the same repeatedly. For example: A pager always functions as a pager.
* **Tightly constrained** − All computing systems have constraints on design metrics, but those on an embedded system can be especially tight. Design metrics is a measure of an implementation's features such as its cost, size, power, and performance. It must be of a size to fit on a single chip, must perform fast enough to process data in real time and consume minimum power to extend battery life.
* **Reactive and Real time** − Many embedded systems must continually react to changes in the system's environment and must compute certain results in real time without any delay. Consider an example of a car cruise controller; it continually monitors and reacts to speed and brake sensors. It must compute acceleration or de-accelerations repeatedly within a limited time; a delayed computation can result in failure to control of the car.
* **Microprocessors based** − It must be microprocessor or microcontroller based.
* **Memory** − It must have a memory, as its software usually embeds in ROM. It does not need any secondary memories in the computer.
* **Connected** − It must have connected peripherals to connect input and output devices.
* **HW-SW systems** − Software is used for more features and flexibility. Hardware is used for performance and security.

## **Basic Structure of an Embedded System**

The following illustration shows the basic structure of an embedded system:



* **Sensor** − It measures the physical quantity and converts it to an electrical signal which can be read by an observer or by any electronic instrument like an A2D converter. A sensor stores the measured quantity to the memory.
* **A-D Converter** − An analog-to-digital converter converts the analog signal sent by the sensor into a digital signal.
* **Processor & ASICs** − Processors process the data to measure the output and store it to the memory.
* **D-A Converter** − A digital-to-analog converter converts the digital data fed by the processor to analog data
* **Actuator** − An actuator compares the output given by the D-A Converter to the actual (expected) output stored in it and stores the approved output.

## **Processors**

Processor is the heart of an embedded system. It is the basic unit that takes inputs and produces an output after processing the data. For an embedded system designer, it is necessary to have the knowledge of both microprocessors and microcontrollers.

A processor has two essential units

* Program Flow Control Unit (CU)
* Execution Unit (EU)

The CU includes a fetch unit for fetching instructions from the memory. The EU has circuits that implement the instructions pertaining to data transfer operation and data conversion from one form to another.

The EU includes the Arithmetic and Logical Unit (ALU) and also the circuits that execute instructions for a program control task such as interrupt, or jump to another set of instructions.

A processor runs the cycles of fetch and executes the instructions in the same sequence as they are fetched from memory.

## **Microcontroller**

A microcontroller is a single-chip VLSI unit (that is also called **microcomputer**), although it have limited computational capabilities, possesses enhanced input/output capability and a number of on-chip functional units.

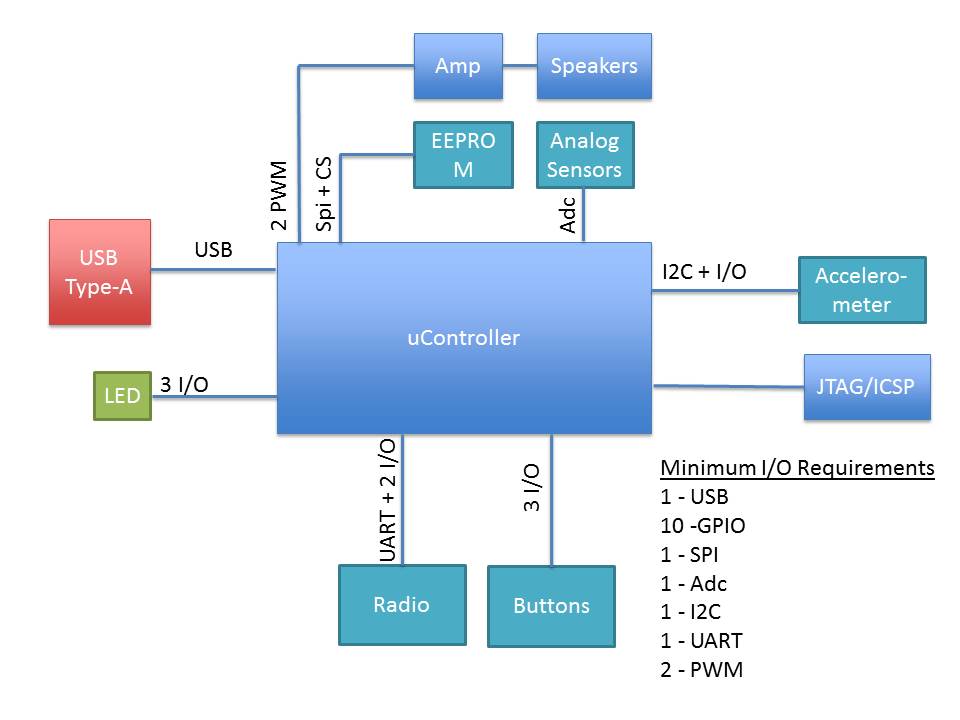
Microcontrollers are particularly used in embedded systems for real-time control applications with on-chip program memory and devices.

### **2.1.5.1 How to choose microcontroller**

Selecting the right microcontroller for a product can be a daunting task.  Not only are there a number of technical features to consider, there are also business case issues such as cost and lead-times that can cripple a project.  At the start of a project there is a great temptation to jump in and start selecting a microcontroller before the details of the system has been hashed out.  This is of course a bad idea.  Before any thought is given to the microcontroller, the hardware and software engineers should work out the high levels of the system, block diagram and flowchart them and only then is there enough information to start making a rational decision on microcontroller selection.    When that point is reached, there are 10 easy steps that can be followed to ensure that the right choice is made.

* **Step 1: Make a list of required hardware interfaces**

Using the general hardware block diagram, make a list of all the external interfaces that the microcontroller will need to support.  There are two general types of interfaces that need to be listed.  The first are communication interfaces.  These are peripherals such as USB, I2C, SPI, UART, and so on.  Make a special note if the application requires USB or some form of Ethernet.  These interfaces greatly affect how much program space the microcontroller will need to support.  The second type of interface is digital inputs and outputs, analog to digital inputs, PWM’s, etc.  These two interface types will dictate the number of pins that will be required by the microcontroller.  Figure 2-2 shows a generic example of a block diagram with the i/o requirements listed.



**Figure 2-2 | List of Hardware Features**

* **Step 2: Examine the software architecture**

The software architecture and requirements can greatly affect the selection of a microcontroller. How heavy or how light the processing requirements will determine whether you go with an 80 MHz DSP or an 8 MHz 8051.  Just like with the hardware, make notes of any requirements that will be important. For example, do any of the algorithms require floating point mathematics?  Are there any high frequency control loops or sensors?  Estimate how long and how often each task will need to run.  Get an order of magnitude feel for how much processing power will be needed.  The amount of computing power required will be one of the biggest requirements for the architecture and frequency of the microcontroller.

* **Step 3: Select the architecture**

Using the information from steps 1 and 2 an engineer should be able to start getting an idea of the architecture that will be needed.  Can the application get by with eight bit architectures?  How about 16 bits?  Does it require a 32 bit ARM core?  Between the application and the required software algorithms these questions will start to converge on a solution.  Don’t forget keep in mind possible future requirements and feature creep.  Just because you could currently get by with an 8 bit microcontroller doesn’t mean you shouldn’t consider a 16 bit microcontroller for future features or even for ease of use.  Don’t forget that microcontroller selection can be an iterative process. You may select a 16-bit part in this step but then in a later step find that a 32 bit ARM part works better. This step is simply gets an engineer to look in the right direction.

* **Step 4: Identify Memory Needs**

Flash and RAM are two very critical components of any microcontrollers.  Making sure that you don’t run out of program space or variable space is undoubtedly of highest priority.  It is far easier to select a part with too much of these features than not enough.  Getting to the end of a design and discovering that you need 110% or that features need to be cut just isn’t going to fly.  After all, you can always start with more and then later move to a more constrained part within the same chip family.  Using the software architecture and the communication peripherals included in the application, an engineer can estimate how much flash and RAM will be required for the application.  Don’t forget to leave room for feature creep and the next versions!  It will save many headaches in the future.

* **Step 5: Start searching for microcontrollers**

Now that there is a better idea of what the required features of the microcontroller will be the search can begin!  One place that can be a good place to start is with a microcontroller supplier such as Arrow, Avnet, Future Electronics or similar.  Talk with an FAE about your application and requirements and often times they can direct you to a new part that is cutting edge and meets the requirements.  Just keep in mind that they might have pressure on them at that time to push a certain family of microcontrollers!

The next best place to start is with a silicon provider that you are already familiar with.  For example, if you have used Microchip parts in the past and had a good experience with them, then start at their website. Most silicon providers have a search engine that allows you to enter your peripheral sets, I/O and power requirements and it will narrow down the list of parts that match the criteria. From that list the engineer can then move forward towards selecting a microcontroller.

* **Step 6: Examine Costs and Power Constraints**

At this point the selection process has revealed a number of potential candidates.  This is a great time to examine the power requirements and cost of the part.  If the device will be powered from a battery and mobile, then making sure the parts are low-power is absolutely precarious.  If it doesn’t meet power requirements then keep weeding the list down until you have a select few.  Don’t forget to examine the piece price of the processor either.  While prices have steadily been approaching $1 in volume for many parts, if it is highly specialized or a high-end processing machine then price might be critical.  Don’t forget about this key element.

* **Step 7: Check part availability**

With the list of potential parts in hand, now is a good time to start checking on how available the part is. Some of the things to keep in mind are what the lead times for the part?  Are they kept in stock at multiple distributors or is there 6 – 12 week lead time? What are your requirements for availability?  You don’t want to get stuck with a large order and have to wait three months to be able to fill it.  Then there is a question of how new the part is and whether it will be around for the duration of your product life cycle.  If your product will be around for 10 years then you need to find a part that the manufacturer guarantees will still be built in 10 years.

* **Step 8: Select a development kit**

One of the best parts of selecting a new microcontroller is finding a development kit to play with and learn the inner working of the controller.  Once an engineer has settled their heart on the part they want to use they should research what development kits are available.  If a development kit isn’t available then the selected part is most likely not a good choice and they should go back a few steps and find a better part.  Most development kits today cost under $100.  Paying any more than that (unless it is designed to work with multiple processor modules) is just too much.  Another part may be a better   choice.

* **Step 9: Investigate compilers and tools**

The selection of the development kit nearly solidifies the choice of microcontroller.  The last consideration is to examine the compiler and tools that are available.  Most microcontrollers have a number of choices for compilers, example code and debugging tools.  It is important to make sure that all the necessary tools are available for the part.

* **Step 10: Start Experimenting**

Even with the selection a microcontroller nothing is set in stone.  Usually the development kit arrives long before the first prototyped hardware.  Take advantage by building up test circuits and interfacing them to the microcontroller. Choose high risk parts and get them working on the development kit.  It may be that you discover the part you thought would work great has some unforeseen issue that would force a different microcontroller to be selected.

### **2.1.5.2 Comparison between different microcontrollers**

Here are the main difference between AVR, ARM, 8051 and PIC which are the most well-known microcontrollers:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **8051** | **PIC** | **AVR** | **ARM** |
| **Bus width** | 8-bit for standard core | 8/16/32-bit | 8/32-bit | 32-bit mostly also available in 64-bit |
| **Communic-ation Protocols** | UART, USART,SPI,I2C | PIC, UART, USART, LIN, CAN, Ethernet, SPI, I2S | UART, USART, SPI, I2C, (special purpose AVR support CAN, USB, Ethernet) | UART, USART, LIN, I2C, SPI, CAN, USB, Ethernet, I2S, DSP, SAI (serial audio interface), IrDA |
| **Speed** | 12 Clock/instruction cycle | 4 Clock/instruction cycle | 1 clock/  instruction cycle | 1 clock/ instruction cycle |
| **Memory** | ROM, SRAM, FLASH | SRAM, FLASH | Flash, SRAM, EEPROM | Flash, SDRAM, EEPROM |
| **ISA** | CLSC | Some feature of RISC | RISC | RISC |
| **Memory Architecture** | Von Neumann architecture | Harvard architecture | Modified | Modified Harvard architecture |
| **Power Consumption** | Average | Low | Low | Low |
| **Families** | 8051 variants | PIC16,PIC17, PIC18, PIC24, PIC32 | Tiny, Atmega, Xmega, special purpose AVR | ARMv4,5,6,7 and series |
| **Community** | Vast | Very Good | Very Good | Vast |
| **Manufacturer** | NXP, Atmel, Silicon Labs, Dallas, Cyprus, Infineon, etc. | Microchip Average | Atmel | Apple, Nvidia, Qualcomm, Samsung Electronics, and TI etc. |
| **Cost  (compared to features provide)** | Very Low | Average | Average | Low |
| **Other Feature** | Known for its Standard | Cheap | Cheap, effective | High speed operation  Vast |
| **Popular Microcontrollers** | AT89C51, P89v51, etc. | PIC18fXX8, PIC16f88X, PIC32MXX | Atmega8, 16, 32, Arduino Community | LPC2148, ARM Cortex-M0 to ARM Cortex-M7, etc. |

Thus, this is all about the difference between AVR, ARM, 8051 and PIC microcontrollers. We hope that you have got a better understanding of this concept.

## **Arduino vs Raspberry Pi**



When embarking on IoT projects, you have to choose the best platform to build your application. People often ask which platforms they should use for what kinds of projects. The answer does depend on what your goals are for your IoT project and what you plan for it to do. In this article, we will discuss when it is best to use Arduino and when to use Raspberry Pi, in some cases, we’ll discuss when you might even want to incorporate both devices into your solution.

### 

### **2.1.6.1 Development environment / tools**

With the Arduino’s limited program storage and very small SRAM the efficiency of your application is important. For that reason, applications are generally written in C/C++ using the Arduino IDE. The applications written for the Arduino are rather appropriately called Sketches because they are generally of a very short nature. Their focus is on interacting with the hardware devices you’ve connected to the Arduino and perhaps transmitting data or receiving commands from another computer.

As stated earlier the Raspberry Pi is a full computer. While the commonly promoted language for the Raspberry Pi is Python, the flexibility of this platform lets user use a variety of languages including Ruby, PHP, and Java. As I’ve stated in an earlier blog post you can run Node JS on the Raspberry Pi and develop IoT apps in Javascript with the Johnny-Five framework and even do visual Node development using NodeRed entirely on the Raspberry Pi.

**2.1.6.2 Sensor connectivity**

The Arduino provides a complete set of 14 digital and 6 analog inputs and outputs. That lets you turn items on and off (digital) and detect digital changes in state of connected components. Unlike the Raspberry Pi the Arduino also has a complete set of analog inputs. These can be used to measure applied voltages to respond to analog changes in things like temperature, light levels, etc. For cases where you need to apply analog output the Arduino also supports pulse-width-modulation (PWM) which is a means of setting an analog level on a component (like the brightness of an LED). In addition to the the array of inputs and outputs the Arduino also has a hardware design that allows add-on boards, called shields, to be coupled to it. These extend the basic, built-in I/O capabilities to controls things like motors, or add network capabilities.

The Raspberry Pi also provides 8 I/O pins, but they are all digital in nature. If you wish to interact with analog devices you’ll need to wire up an additional chip to the digital pins to interact with those devices. A common means of doing this is through a chip called the MCP3008 which costs under $4. There are also a selection of boards for the Raspberry Pi sometimes referred to as HAT (Hardware Attached on Top) which are similar in overall concept the Arduino shields.

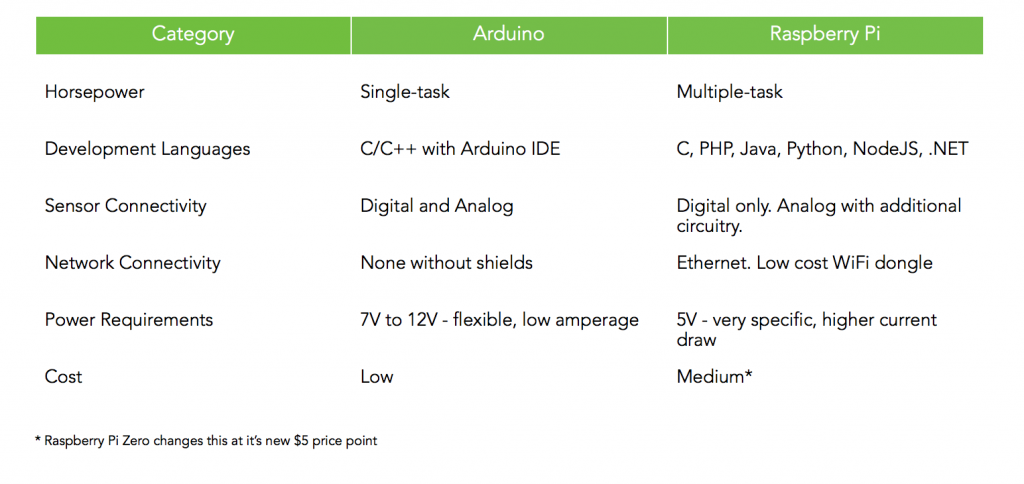
**2.1.6.3 Network Connectivity**

This is an area where the Raspberry Pi has a big lead. The majority of Raspberry Pi boards come with an Ethernet connector right on the board. The USB ports found on the Raspberry Pi also makes it easy to hook a Wi-Fi dongle up to the Raspberry Pi and obtain network connectivity that way.

The Arduino provides no built-in network connectivity except in higher priced Arduino boards like the Arduino Yun. For Arduino projects that require network connectivity various shields are available to support either ethernet or WiFi connectivity

### **2.1.6.4 Which to use**

There is no simple answer. It really depends on the requirements of your project. I’ve summarized some of the decision points in the table below. The IOT device market is quite strong right now and new boards appear on the market frequently. New entrants, like the Raspberry Pi Zero, may shift your decision point in the months to come. One thing is for certain, this is an exciting and dynamic market.



# **2.2 The Web**

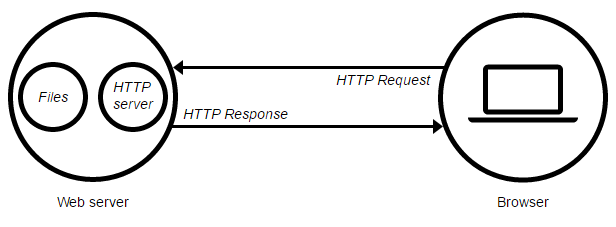
## **HTTP Protocol**

HTTP stands for Hypertext Transfer Protocol. It's a stateless, application-layer protocol for communicating between distributed systems, and is the foundation of the modern web. As a web developer, we all must have a strong understanding of this protocol.

### **2.2.1.1 HTTP Basics**

HTTP allows for communication between a variety of hosts and clients, and supports a mixture of network configurations.

This makes HTTP a stateless protocol. The communication usually takes place over TCP/IP (Transmission Control Protocol/Internet Protocol), but any reliable transport can be used. The default port for TCP/IP is 80, but other ports can also be used.



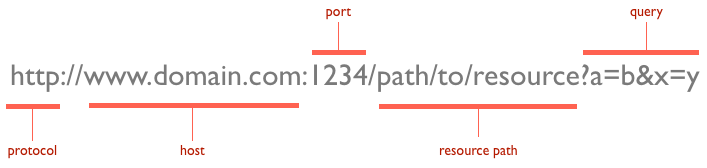
Custom headers can also be created and sent by the client.

Communication between a host and a client occurs, via a request/response pair. The client initiates an HTTP request message, which is serviced through a HTTP response message in return. We will look at this fundamental message-pair in the next section.

The current version of the protocol is HTTP/1.1, which adds a few extra features to the previous 1.0 version. The most important of these, in my opinion, includes persistent connections, chunked transfer-coding and fine-grained caching headers. We'll briefly touch upon these features in this article; in-depth coverage will be provided in part two.

### **2.2.1.2 URLs**

At the heart of web communications is the request message, which are sent via Uniform Resource Locators (URLs). I'm sure you are already familiar with URLs, but for completeness sake, I'll include it here. URLs have a simple structure that consists of the following components:



The protocol is typically http, but it can also be https for secure communications. The default port is 80, but one can be set explicitly, as illustrated in the above image. The resource path is the local path to the resource on the server.

### **2.2.1.3 Verbs**

URLs reveal the identity of the particular host with which we want to communicate, but the action that should be performed on the host is specified via HTTP verbs. Of course, there are several actions that a client would like the host to perform. HTTP has formalized on a few that capture the essentials that are universally applicable for all kinds of applications.

These request verbs are:

* GET: fetch an existing resource. The URL contains all the necessary information the server needs to locate and return the resource.
* POST: create a new resource. POST requests usually carry a payload that specifies the data for the new resource.
* PUT: update an existing resource. The payload may contain the updated data for the resource.
* DELETE: delete an existing resource.

The above four verbs are the most popular, and most tools and frameworks explicitly expose these request verbs. PUT and DELETE are sometimes considered specialized versions of the POST verb, and they may be packaged as POST requests with the payload containing the exact action: create, update or delete.

## **Web Servers**

A web server is a computer system that processes requests via [HTTP](https://en.wikipedia.org/wiki/HTTP), The primary function of a web server is to store, process and deliver [web pages](https://en.wikipedia.org/wiki/Web_page) to [clients](https://en.wikipedia.org/wiki/Client_(computing)).

### **2.2.2.1 Web server’s definition**

A web server can refer to hardware or software, or both of them working together.

1. On the hardware side, a web server is a computer that stores a website's component files (e.g. HTML documents, images, CSS stylesheets, and JavaScript files, not covered in this report) and delivers them to the end-user's device. It is connected to the Internet and can be accessed through a domain name like mozilla.org.
2. On the software side, a web server includes several parts that control how web users access hosted files, at minimum an HTTP server. An HTTP server is a piece of software that understands [URLs](https://developer.mozilla.org/en-US/docs/Glossary/URL) and [HTTP](https://developer.mozilla.org/en-US/docs/Glossary/HTTP).

At the most basic level, whenever a browser needs a file hosted on a web server, the browser requests the file via HTTP. When the request reaches the correct web server (hardware), the HTTP server (software) sends the requested document back, also through HTTP.



### **2.2.2.2 Types of web servers:**

To publish a website, you need either a static or a dynamic web server.

A static web server, or stack, consists of a computer (hardware) with an HTTP server (software). We call it "static" because the server sends its hosted files "as-is" to your browser.

A dynamic web server consists of a static web server plus extra software, most commonly an application server and a database. We call it "dynamic" because the application server updates the hosted files before sending them to your browser via the HTTP server.

For example, to produce the final webpages you see in the browser, the application server might fill an HTML template with contents from a database. Sites like Wikipedia have many thousands of webpages, but they aren't real HTML documents, only a few HTML templates and a giant database. This setup makes it easier and quicker to maintain and deliver the content.

### **2.2.2.3 Famous web servers:**

There are 4 primary web servers:

* [Apache](https://httpd.apache.org/) (provided by Apache)
* [IIS](http://www.iis.net/) (provided by Microsoft and short for Internet Information Server)
* [nginx](https://www.nginx.com/) (provided by NGINX, Inc. and pronounced like “Engine X”)
* and [GWS](https://en.wikipedia.org/wiki/Google_platform#Software) (provided by Google and short for Google Web Server)

Currently, Apache is the most popular with IIS gaining in popularity soon becoming the most popular web server. nginx is an extremely popular alternative as it is very fast and very lightweight, while GWS is the least used with a small percentage of use.

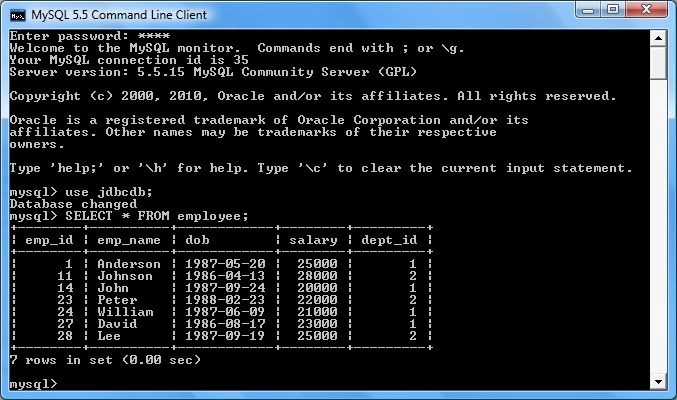
## **Databases**

Organized collection of data is called database. It is really important to organize data and different database model uses unique processes to store large data sets using processes requiring information. For example, the availability of a table in a hotel can be measured by the vacancies.

**2.2.3.1 Database management systems**

Database management systems (DBMSs) are computer software applications that interact with the user define programs, applications, and metadata to capture and analyze data. A general management purpose DBMS is designed to allow structure and design to any data by using some administration protocols of databases.

Some of the well-known DBMSs include MySQL, PostgreSQL, Microsoft SQL Server, Oracle, SAP and DB2. So, a database is a place to store and retrieve organized data. In practical usage, web hosting, databases store your personal blog data, the date they were live, the opinions people posted with them, and your administrative information. It’s all stored in the database classified according to the database model that they support.



using program. According to its characteristics SQLite is a popular choice as an embedded database for local/client storage in application software such as different web browsers. It is perhaps the most widely deployed database server engine used today not only by all leading browsers, operating systems but also by embedded systems, among others.

* MS Access

A database management system from Microsoft – MS Access (Microsoft Access) is the first choice of Software developers, data architects to develop application software because it supports different objects, including DAO (Data Access Objects), ActiveX Data Objects, and many other ActiveX components. MS Access has relation with Microsoft Jet Database Engine to deploy a graphical user interface. It stores data in its own format based on the Access Jet Database Engine. It can also import or link directly to data stored in other applications and databases.

**LAMP Stack**

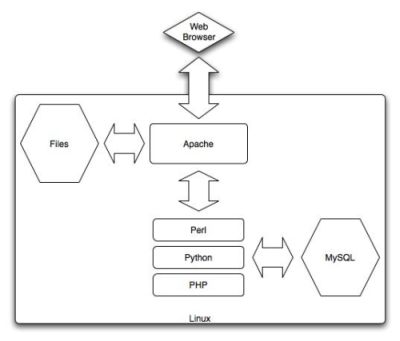
LAMP is an archetypal model of web service [stacks](https://en.wikipedia.org/wiki/Solution_stack), named as an [acronym](https://en.wikipedia.org/wiki/Acronym) of the names of its original four [open-source](https://en.wikipedia.org/wiki/Open-source) components: the [Linux](https://en.wikipedia.org/wiki/Linux) [operating system](https://en.wikipedia.org/wiki/Operating_system), the [Apache HTTP Server](https://en.wikipedia.org/wiki/Apache_HTTP_Server), the [MySQL](https://en.wikipedia.org/wiki/MySQL) [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system) (RDBMS), and the [PHP](https://en.wikipedia.org/wiki/PHP) [programming language](https://en.wikipedia.org/wiki/Programming_language). The LAMP components are largely interchangeable and not limited to the original selection. As a solution stack, LAMP is suitable for building [dynamic web sites](https://en.wikipedia.org/wiki/Dynamic_web_site) and [web applications](https://en.wikipedia.org/wiki/Web_application).

**2.2.4.1 Exploding the Acronym**

Simply exploding the acronym on a letter by letter basis gives us the following elements:

* Linux
* Apache Web server
* MySQL database
* Perl, Python, or PHP

Individually, each of these items is a powerful component in its own right. The key to the idea behind LAMP, a term originally coined by Michael Kunze in the German magazine c't in 1998, is the use of these items together. Although not actually designed to work together, these open source software alternatives are readily and freely available. This has lead to them often being used together. In the past few years, their compatibility and use together has grown and been extended. Certain extensions have even been created specifically to improve the cooperation between different components.



Each of the components in the LAMP stack is an example of Free or Open Source Software (FOSS). The benefit of the FOSS approach is three-fold. First, the nature of FOSS software means applications are available for free download, making them readily available to a wide range of people without payment. That makes the software incredibly attractive to a wide range of users who would otherwise have to pay for "professional" commercial tools, which is often an expensive step in producing a Web site.

Second, FOSS licenses are open and thus have few restrictions on their use and deployment of applications based on the FOSS technology. It is possible to develop and deploy LAMP-based projects without paying any license fees for distributing the software, and this, again, makes it popular for both hobbyists and professionals alike.

Third, and a major reason for the growth and use of FOSS technology (including LAMP), is that because users have access to the source it is much easier to fix faults and improve the applications. In combination with the open license, this simplifies the development process for many enterprises and gives them flexibility that simply isn't available within the confines of a proprietary or commercial-based product.

**2.2.4.2 Using the LAMP Stack**

Choosing to use LAMP in your business is not about cost — although many enterprises will be attracted to the low-cost required for both development and deployment. Instead, choosing LAMP for your organization is about the benefits it provides, as summarized below.

* Flexibility: There are no limits to what you can do with the LAMP stack, either technically or because of licensing restrictions. This allows you the flexibility to build and deploy applications in a method that suits you, not the supplier of the technology you are using.
* Customization: Because LAMP components are open source, they have built up a huge array of additional components and modules that provide additional functionality. The open source approach enables you to do the same, customizing components and functionality to suit your needs.
* Ease of Development: You can write powerful applications using LAMP technology in relatively few lines of code. Often the code is straightforward enough that even nonprogrammers can modify or extend the application.
* Ease of Deployment: With neither licensing issues nor the need to compile applications, deployment is often as easy as copying an application to a new host. Most hosting services provide LAMP-based environments as standard, or they can be deployed using a Linux distribution, such as Fedora or Debian.
* Security: With many eyes developing the software and years of use by a wide range of users and community groups, LAMP technology is secure and stable. Problems are normally fixed very quickly, and without the need for a costly support contract.
* Community and Support: A wide and experienced group of people is willing to provide help and support during the development and deployment of LAMP-based applications.

Many successful business have already leveraged the use of LAMP technology. Many heavily trafficked Web sites use LAMP, or components of it, to support their applications.

# **2.3 The Project**

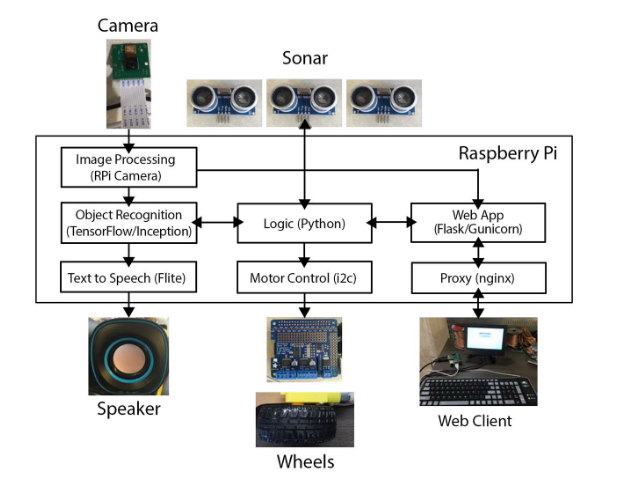
Object recognition is one of the most exciting areas in machine learning right now. Computers have been able to recognize objects like faces or cats reliably for quite a while, but recognizing arbitrary objects within a larger image has been the Holy Grail of artificial intelligence. Maybe the real surprise is that human brains recognize objects so well. We effortlessly convert photons bouncing off objects at slightly different frequencies into a spectacularly rich set of information about the world around us. Machine learning still struggles with these simple tasks, but in the past few years, it’s gotten much better.

Deep learning and a large public training data set called ImageNet has made an impressive amount of progress toward object recognition. TensorFlow is a well-known framework that makes it very easy to implement deep learning algorithms on a variety of architectures. TensorFlow is especially good at taking advantage of GPUs, which in turn are also very good at running deep learning algorithms.

**Building the robot**

The new third generation Raspberry Pi is perfect for this kind of project. Bit costly but with WiFi, a quad core CPU, and a gigabyte of RAM. And maybe adding a microSD card that can load Rasbian within, which is basically Debian. The result is a perfect combination for that robot.

Getting to DC motors interfaced with raspberry pi, being run at a higher current than the Raspberry Pi can provide, so a separate controller is necessary, an Adafruit motor hat would surely be a super convenient. The hardware is extremely forgiving, and Adafruit provides a nice library and tutorial to control the motors over [I2C](https://en.wikipedia.org/wiki/I%C2%B2C). Initially, it’s cheaper to use motor controllers, but it’s optional to order a better quality replacement.



A [camera](https://www.amazon.com/Arducam-Megapixels-Sensor-OV5647-Raspberry/dp/B012V1HEP4/ref=sr_1_2?s=electronics&ie=UTF8&qid=1473286363&sr=1-2&keywords=raspberry+pi+camera) attaches right into the Raspberry Pi and provides a real-time video feed that may recognize objects.

The Raspberry Pi needs about 2 amps of current, but 3 amps is safer with the speaker we’re going to plug into it. It’s necessary to select a battery whose chargers work awesomely for this task. Small chargers don’t actually output enough amps and can cause problems.

A couple of [HC-SR04 sonar sensors](https://www.amazon.com/CJRSLRB%C2%AE-Ultrasonic-Measuring-Transducer-Duemilanove/dp/B016XJABP0/ref=sr_1_3?s=automotive&ie=UTF8&qid=1473286270&sr=1-3&keywords=hc+sr04) help the robot avoid crashing into things.

Cheap USB speakers can be added.

**Programming the robot**

First it’s required to install TensorFlow, TensorFlow actually comes with a makefile that lets you build it right on the system.

TensorFlow comes with a prebuilt model called “inception” that performs object recognition.

The model works surprisingly well on a wide range of inputs, but it’s clearly missing an accurate “prior,” or a sense of what things it’s likely to see, and there are quite a lot of objects missing from the training data.