**CHAPTER 1**

**OVERVIEW OF SENSORS**

**1.1 Outline**

For embedded systems, sensors form an important part. A sensor is a device that measures a physical quantity and converts it into a signal which can be read by an observer or by an instrument. For example, a mercury-in-glass thermometer converts the measured temperature into expansion and contraction of a liquid which can be read on a calibrated glass tube. A thermocouple converts temperature to an output voltage which can be read by a voltmeter. For accuracy, all sensors need to be calibrated against known standards.

Sensors are used in everyday objects such as touch-sensitive elevator buttons (tactile sensor) and lamps which dim or brighten by touching the base. There are also innumerable applications for sensors of which most people are never aware. Applications include cars, machines, aerospace, medicine, manufacturing and robotics.

A sensor is a device which receives and responds to a signal or stimulus. Here, the term "stimulus" means a property or a quantity that needs to be converted into electrical form. Hence, sensor can be defined as a device which receives a signal and converts it into electrical form which can be further used for electronic devices. A sensor differs from a transducer in the way that a transducer converts one form of energy into other form whereas a sensor converts the received signal into electrical form only.

A sensor's sensitivity indicates how much the sensor's output changes when the measured quantity changes. For instance, if the mercury in a thermometer moves 1 cm when the temperature changes by 1 °C, the sensitivity is 1 cm/°C. Sensors that measure very small changes must have very high sensitivities. Sensors also have an impact on what they measure; for instance, a room temperature thermometer inserted into a hot cup of liquid cools the liquid while the liquid heats the thermometer. Sensors need to be designed to have a small effect on what is measured, making the sensor smaller often improves this and may introduce other advantages. Technological progress allows more and more sensors to be manufactured on a microscopic scale as microsensors using MEMS technology. In most cases, a microsensor reaches a significantly higher speed and sensitivity compared with macroscopic approaches.

A good sensor obeys the following rules:

* Is sensitive to the measured property
* Is insensitive to any other property likely to be encountered in its application
* Does not influence the measured property

**Choosing a Sensor**:

Factors to consider when choosing a sensor-

* Accuracy - The statistical variance about the exact reading.
* Calibration - Required for most measuring systems since their readings will drift over time.
* Cost
* Environmental - Sensors typically have temperature and/or humidity limits.
* Range - Limits of measurement or the sensor.
* Repeatability - The variance in a sensor's reading when a single condition is repeatedly measured .
* Resolution - The smallest increment the sensor can detect.

**Statistics about sensors**:

MARKETS: Highest demand and growth sector for sensors worldwide are motor vehicles. Process industries are second in demand. Excellent prospects for sensors in telecommunication.

COUNTRIES: Western Europe has 31.8% of the global demand at present; the USA, 31.2% and Japan, 19.4%. The share of other countries will grow from 17.7% in 1998 to 19% by 2008

TRENDS: Sensors on semiconductor basis will increase their market share from 38.9% in 1998 to 43% in 2008. Strong growth expected for sensors based on MEME-technologies, smart sensors and sensors wit bus capabilities.

**1.2 Type of Sensors and Applications**

**1.2.1 Type of Sensors**

Electrical and mechanical sensors are widely used to characterize the performance and properties of components and systems, but are also found in household objects.

Sensors are generally categorized by the type of phenomenon that they measure, rather than the functionality of the sensor itself.

**Mechanical Sensors**

Mechanical sensors measure a property through mechanical means, although the measurement itself may be collected electronically. An example of a mechanical sensor is a strain gauge. The strain gauge measures the physical deformation of a component by experiencing the same strain as the component, yet the change in resistance of the strain gauge is measured electrically. Other types of mechanical sensors include:

* **Pressure sensors**: A pressure sensor measures pressure, typically of gases or liquids. Pressure is an expression of the force required to stop a fluid from expanding, and is usually stated in terms of force per unit area. A pressure sensor usually acts as a transducer; it generates a signal as a function of the pressure imposed. For the purposes of this article, such a signal is electrical. Pressure sensors are used for control and monitoring in thousands of everyday applications. Pressure sensors can also be used to indirectly measure other variables such as fluid/gas flow, speed, water level, and altitude
* **Accelerometer**s: An accelerometer is a device that measures proper acceleration, the acceleration experienced relative to freefall. Single and multi-axis models are available to detect magnitude and direction of the acceleration as a vector quantity, and can be used to sense position, vibration and shock. Micro-machined accelerometers are increasingly present in portable electronic devices and video game controllers, to detect the position of the device or provide for game input.
* **Potentiometers:** A potentiometer (colloquially known as a "pot") is a three-terminal resistor with a sliding contact that forms an adjustable voltage divider. If only two terminals are used (one side and the wiper), it acts as a variable resistor or rheostat. Potentiometers are commonly used to control electrical devices such as volume controls on audio equipment
* **Gas and fluid flow meters**: A gas meter is used to measure the volume of fuel gases such as natural gas and propane. Gas meters are used at residential, commercial, and industrial buildings that consume fuel gas supplied by a gas utility.Fluid Flow measurement is the quantification of bulk fluid movement For liquids, various units are used depending upon the application and industry, but might include gallons (U.S. liquid or imperial) per minute, liters per second, bushels per minute or, when describing river flows, cumecs (cubic metres per second) or acre-feet per day
* **Humidity sensors**: There are various devices used to measure and regulate humidity. A device used to measure humidity is called a psychrometer or hygrometer. A humidistat is used to regulate the humidity of a building with a de-humidifier. These can be analogous to a thermometer and thermostat for temperature control.

**Electrical Sensors**:

Electrical sensors measure electric and magnetic properties. An example of an electrical sensor is an ohmmeter, which is used to measure electrical resistance between two points in a circuit. An ohmmeter sends a fixed voltage through one probe, and measures the returning voltage through a second probe. The drop in voltage is proportional to the resistance, as dictated by Ohm's Law. Other electrical sensors include:

* **Voltmeter/Ammeter**: A voltmeter is an instrument used for measuring the electrical potential difference between two points in an electric circuit. Analog voltmeters move a pointer across a scale in proportion to the voltage of the circuit; digital voltmeters give a numerical display of voltage by use of an analog to digital converter.

An ammeter is a measuring instrument used to measure the electric current in a circuit. Electric currents are measured in amperes (A), hence the name. Smaller values of current can be measured using a milliameter or a microammeter.

* **Metal detector**: A metal detector is a device which responds to metal that may not be readily apparent. The simplest form of a metal detector consists of an oscillator producing an alternating current that passes through a coil producing an alternating magnetic field. If a piece of electrically conductive metal is close to the coil, eddy currents will be induced in the metal, and this produces an alternating magnetic field of its own. If another coil is used to measure the magnetic field (acting as a magnetometer), the change in the magnetic field due to the metallic object can be detected.
* **Radar**:Radar is an object detection system that uses electromagnetic waves to identify the range, altitude, direction, or speed of both moving and fixed objects such as aircraft, ships, motor vehicles, weather formations, and terrain. The term *RADAR* was coined in 1940 by the U.S. Navy as an acronym for RAdio Detection And Ranging.
* **Magnetometer**: A magnetometer is a scientific instrument used to measure the strength and/or direction of the magnetic field in the vicinity of the instrument. Magnetism varies from place to place and differences in Earth's magnetic field (the magnetosphere) can be caused by the differing nature of rocks and the interaction between charged particles from the Sun and the magnetosphere of a planet. Magnetometers are a frequent component instrument on spacecraft that explore planets.

**Thermal Sensors:**

Although all thermal sensors measure changes in temperature, there are a variety of types of thermal sensors, each with specific uses, temperature ranges, and accuracies. Some types of thermal sensors include:

* **Thermometers**: A thermometer (from the Greek *θερμός*(*thermo*) meaning "warm" and *meter*, "to measure") is a device that measures temperature or temperature gradient using a variety of different principles
* **Thermocouples**: A thermocouple is a junction between two different metals that produces a voltage related to a temperature difference. Thermocouples are a widely used type of temperature sensor for measurement and control and can also be used to convert heat into electric power.
* **Thermistors**: A thermistor is a type of resistor whose resistance varies with temperature. The word is a portmanteau of thermal and resistor. Thermistors are widely used as inrush current limiters, temperature sensors, self-resetting overcurrent protectors, and self-regulating heating elements.

**Chemical Sensors:**

Chemical sensors generally detect the concentration of a substance in the air or in a liquid. Some chemical sensors, such as pH glass electrodes are designed to be sensitive to a certain ion. Some other types of chemical sensors include:

* **Oxygen sensors**: An oxygen sensor, or lambda sensor, is an electronic device that measures the proportion of oxygen (O2) in the gas or liquid being analyzed.
* **Carbon monoxide detectors**: A carbon monoxide detector or CO detector is a device that detects the presence of the carbon monoxide (CO) gas in order to prevent carbon monoxide poisoning. CO is a colorless and odorless compound produced by incomplete combustion
* **Redox electrodes**: The redox electrode, is the electrode in an electrochemical system on which the reaction of interest is occurring. The working electrode is often used in conjunction with an auxiliary electrode, and a reference electrode in a three electrode system.

**Optical Sensors:**

Optical sensors detect the presence of light waves. This could include light in the visible spectrum, or outside the visible spectrum, in the case of infrared sensors. Some types of optical sensors include:

* **Photodetectors**: Photosensors or photodetectors are sensors of light or other electromagnetic energy
* **Infrared sensors**: Infrared camera, is a device that forms an image using infrared radiation, similar to a common camera that forms an image using visible light. Instead of the 450–750 nanometer range of the visible light camera, infrared cameras operate in wavelengths as long as 14,000 nm (14 µm).
* **Fiberoptic sensors**: A fiber optic sensor is a sensor that uses optical fiber either as the sensing element ("intrinsic sensors"), or as a means of relaying signals from a remote sensor to the electronics that process the signals ("extrinsic sensors"). Fibers have many uses in remote sensing
* **Interferometers**: Interferometry is the technique of diagnosing the properties of two or more waves by studying the pattern of interference created by their superposition. The instrument used to interfere the waves together is called an interferometer. Interferometry is an important investigative technique in the fields of astronomy, fibre optics, engineering metrology, optical metrology, oceanography, seismology, quantum mechanics, nuclear and particle physics, plasma physics, and remote sensing.

**Other Types of Sensors:**

Latest developments in technology have given rise to many new types of sensors. Some of them are:

* **Nanosensors:** Nanosensors are any biological or chemical sensory points used to convey information about [nanoparticles](http://en.wikipedia.org/wiki/Nanoparticles) to the [macroscopic](http://en.wikipedia.org/wiki/Macroscopic) world. Their use mainly include various medicinal purposes and as gateways to building other nanoproducts, such as computer chips that work at the nanoscale and [nanorobots](http://en.wikipedia.org/wiki/Nanorobot). Presently, there are several ways proposed to make nanosensors, including [top-down lithography, bottom-up assembly](http://en.wikipedia.org/wiki/Top-down_and_bottom-up_design#top-down_lithography.2C_bottom-up_assembly), and [molecular self-assembly](http://en.wikipedia.org/wiki/Molecular_self-assembly). Currently, the most common mass-produced functioning nanosensors exist in the biological world as natural receptors of outside stimulation. The application of nanotechnology to sensors should allow improvements in functionality. In particular, new biosensor technology combined with micro and nanofabrication technology can deliver a huge range of applications. They should also lead to much decreased size, enabling the integration of ‘nanosensors’ into many other devices. We can also expect to see actuators that control movement on the nanoscale. Sensor/actuator combinations will deliver ‘smart’ and precise functions in products and processes. For example, nanofabrication and inspection tools require sensors and actuator systems that can position objects with nanometre accuracy. In this way, sensors and actuators constitute another enabling technology.
* **Biosensors:** A biosensor is a device for the detection of an [analyte](http://en.wikipedia.org/wiki/Analyte) that combines a biological component with a physicochemical detector component. It consists of 3 parts (i) the sensitive biological element (biological material (eg. tissue, microorganisms, organelles, cell receptors, [enzymes](http://en.wikipedia.org/wiki/Enzyme), [antibodies](http://en.wikipedia.org/wiki/Antibody), [nucleic acids](http://en.wikipedia.org/wiki/Nucleic_acid), etc), a biologically derived material or biomimic) The sensitive elements can be created by [biological engineering](http://en.wikipedia.org/wiki/Biological_engineering). (ii) the transducer or the detector element (works in a physicochemical way; optical, piezoelectric, electrochemical, etc.) that transforms the signal resulting from the interaction of the analyte with the biological element into another signal (i.e., transducers) that can be more easily measured and quantified; (iii) associated electronics or signal processors that are primarily responsible for the display of the results in a user-friendly way.

A common example of a commercial biosensor is the blood glucose biosensor, which uses the enzyme [glucose oxidase](http://en.wikipedia.org/wiki/Glucose_oxidase) to break blood glucose down. A [canary in a cage](http://en.wikipedia.org/wiki/Domestic_Canary#Miner.27s_canary), as used by miners to warn of gas, could be considered a biosensor. Many of today's biosensor applications are similar, in that they use organisms which respond to [toxic](http://en.wikipedia.org/wiki/Toxic) substances at a much lower concentrations than humans can detect to warn of the presence of the toxin. Such devices can be used in environmental monitoring, trace gas detection and in water treatment facilities.

**1.2.2 Sensor Networks**

Smart environments represent the next evolutionary development step in building, utilities, industrial, home, shipboard, and transportation systems automation. Like any sentient organism, the smart environment relies first and foremost on sensory data from the real world. Sensory data comes from multiple sensors of different modalities in distributed locations. The smart environment needs information about its surroundings as well as about its internal workings; this is captured in biological systems by the distinction between exteroceptors and proprioceptors. PDABSC(Base Station Controller, Preprocessing)BSTWireless Sensor Machine Monitoring, Medical Monitoring, Wireless Sensor, Wireless Data Collection, Networks Wireless(Wi-Fi 802.11 2.4GHzBlueToothCellular Network, -CDMA, GSM)Printer Wireland (Ethernet WLAN, Optical)Animal Monitoring, Vehicle Monitoring, Online monitoring, Server transmitter Anywhere, any time to access Notebook, Cellular, Phone, PC, Ship Monitoring, Wireless Sensor Networks, Roving Human monitor, Data Distribution, Network Management Center(Database large storage, analysis)Data Acquisition Network.

The challenges in the hierarchy of: detecting the relevant quantities, monitoring and collecting the data, assessing and evaluating the information, formulating meaningful user displays, and performing decision-making and alarm functions are enormous. The information needed by smart environments is provided by Distributed Wireless Sensor Networks, which are responsible for sensing as well as for the first stages of the processing hierarchy. The importance of sensor networks is highlighted by the number of recent funding initiatives, including the DARPA SENSIT program, military programs, and NSF Program Announcements.

A sensor network is a group of specialized transducers with a communications infrastructure intended to monitor and record conditions at diverse locations. Commonly monitored parameters are temperature, humidity, pressure, wind direction and speed, illumination intensity, vibration intensity, sound intensity, power-line voltage, chemical concentrations, pollutant levels and vital body functions.

A sensor network consists of multiple detection stations called sensor nodes, each of which is small, lightweight and portable. Every sensor node is equipped with a transducer, microcomputer, transceiver and power source. The transducer generates electrical signals based on sensed physical effects and phenomena. The microcomputer processes and stores the sensor output. The transceiver, which can be hard-wired or wireless, receives commands from a central computer and transmits data to that computer. The power for each sensor node is derived from the electric utility or from a battery.

Potential applications of sensor networks include:

* Industrial automation
* Automated and smart homes
* Video surveillance
* Traffic monitoring
* Medical device monitoring
* Monitoring of weather conditions
* Air traffic control
* Robot control

A sensor node might vary in size from that of a shoebox down to the size of a grain of dust, although functioning "motes" of genuine microscopic dimensions have yet to be created. The cost of sensor nodes is similarly variable, ranging from hundreds of dollars to a few pennies, depending on the size of the sensor network and the complexity required of individual sensor nodes. Size and cost constraints on sensor nodes result in corresponding constraints on resources such as energy, memory, computational speed and bandwidth.

A sensor network normally constitutes a wireless ad-hoc network, meaning that each sensor supports a multi-hop routing algorithm (several nodes may forward data packets to the base station).

**1.2.3 Wireless Sensor Networks**

A wireless sensor network (WSN) consists of spatially distributed autonomous sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance. They are now used in many industrial and civilian application areas, including industrial process monitoring and control, machine health monitoring, environment and habitat monitoring, healthcare applications, home automation, and traffic control.

In computer science and telecommunications, wireless sensor networks are an active research area with numerous workshops and conferences arranged each year. The study of wireless sensor networks is challenging in that it requires an enormous breadth of knowledge from an enormous variety of disciplines.

The figure shows the complexity of wireless sensor networks, which generally consist of a data acquisition network and a data distribution network, monitored and controlled by a management center. The plethora of available technologies makes even the selection of components difficult, let alone the design of a consistent, reliable, robust overall system.

Fig 1.1 Wireless Sensor Networks

**1.2.4 Sensor Networks Applications**

The applications for WSNs are varied, typically involving some kind of monitoring, tracking, or controlling. Specific applications include habitat monitoring, object tracking, nuclear reactor control, fire detection, and traffic monitoring. In a typical application, a WSN is scattered in a region where it is meant to collect data through its sensor nodes.

**Area monitoring**: Area monitoring is a common application of WSNs. In area monitoring, the WSN is deployed over a region where some phenomenon is to be monitored. For example, a large quantity of sensor nodes could be deployed over a battlefield to detect enemy intrusion instead of using landmines. When the sensors detect the event being monitored (heat, pressure, sound, light, electro-magnetic field, vibration, etc), the event needs to be reported to one of the base stations, which can take appropriate action (e.g., send a message on the internet or to a satellite). Depending on the exact application, different objective functions will require different data-propagation strategies, depending on things such as need for real-time response, redundancy of the data (which can be tackled via data aggregation and information fusion techniques), need for security, etc.

**Environmental monitoring**: A number of WSNs have been deployed for environmental monitoring. Many of these have been short lived, often due to the prototype nature of the projects. Examples of longer-lived deployments are monitoring the state of permafrost in the Swiss Alps: The PermaSense Project, PermaSense Online Data Viewer and glacier monitoring.

**Machine Health Monitoring or Condition based maintenance**: Wireless sensor networks have been developed for machinery condition-based maintenance (CBM) as they offer significant cost savings and enable new functionalities. In wired systems, the installation of enough sensors is often limited by the cost of wiring, which runs between $10–$1000 per foot. Previously inaccessible locations, rotating machinery, hazardous or restricted areas, and mobile assets can now be reached with wireless sensors. Often, companies use manual techniques to calibrate, measure, and maintain equipment. This labor-intensive method not only increases the cost of maintenance but also makes the system prone to human errors. Especially in US Navy shipboard systems, reduced manning levels make it imperative to install automated maintenance monitoring systems. Wireless sensor networks play an important role in providing this capability.

**Industrial Monitoring**

**Water/Wastewater Monitoring**: There are many opportunities for using wireless sensor networks within the water/wastewater industries. Facilities not wired for power or data transmission can be monitored using industrial wireless I/O devices and sensors powered using solar panels or battery packs. As part of the American Recovery and Reinvestment Act (ARRA), funding is available for some water and wastewater projects in most states.

**Landfill Ground Well Level Monitoring and Pump Counter**: Wireless sensor networks can be used to measure and monitor the water levels within all ground wells in the landfill site and monitor leachate accumulation and removal. A wireless device and submersible pressure transmitter monitors the leachate level. The sensor information is wirelessly transmitted to a central data logging system to store the level data, perform calculations, or notify personnel when a service vehicle is needed at a specific well.

**Flare Stack Monitoring**: Landfill managers need to accurately monitor methane gas production, removal, venting, and burning. Knowledge of both methane flow and temperature at the flare stack can define when methane is released into the environment instead of combusted. To accurately determine methane production levels and flow, a pressure transducer can detect both pressure and vacuum present within the methane production system.

**Water Tower Level Monitoring**: Water towers are used to add water and create water pressure to small communities or neighborhoods during peak use times to ensure water pressure is available to all users. Maintaining the water levels in these towers is important and requires constant monitoring and control. A wireless sensor network that includes submersible pressure sensors and float switches monitors the water levels in the tower and wirelessly transmits this data back to a control location. When tower water levels fall, pumps to move more water from the reservoir to the tower are turned on.

**Vehicle Detection**: Wireless sensor networks can use a range of sensors to detect the presence of vehicles ranging from motorcycles to train cars.

**Agriculture**: Using wireless sensor networks within the agricultural industry is increasingly common. Gravity fed water systems can be monitored using pressure transmitters to monitor water tank levels, pumps can be controlled using wireless I/O devices, and water use can be measured and wirelessly transmitted back to a central control center for billing. Irrigation automation enables more efficient water use and reduces waste.

**Windrow Composting**: Composting is the aerobic decomposition of biodegradable organic matter to produce compost, a nutrient-rich mulch of organic soil produced using food, wood, manure, and/or other organic material. One of the primary methods of composting involves using windrows. To ensure efficient and effective composting, the temperatures of the windrows must be measured and logged constantly. With accurate temperature measurements, facility managers can determine the optimum time to turn the windrows for quicker compost production. Manually collecting data is time consuming, cannot be done continually, and may expose the person collecting the data to harmful pathogens. Automatically collecting the data and wirelessly transmitting the data back to a centralized location allows composting temperatures to be continually recorded and logged, improving efficiency, reducing the time needed to complete a composting cycle, and minimizing human exposure and potential risk.

**Greenhouse Monitoring**: Wireless sensor networks are also used to control the temperature and humidity levels inside commercial greenhouses. When the temperature and humidity drops below specific levels, the greenhouse manager must be notified via e-mail or cell phone text message, or host systems can trigger misting systems, open vents, turn on fans, or control a wide variety of system responses. Because some wireless sensor networks are easy to install, they are also easy to move as the needs of the application change.

**1.3 Objective**

Energy being the scarce resource that it already is, we set out to build a system that conserves energy and also performs its functions efficiently. The system is capable of detecting the presence of people in the room and automatically switching on the lights and when the last person exits, it automatically switches off. This ensures that electricity is not unnecessarily wasted when people forget to switch off the lights while leaving the room.

Another feature of the system is that we have tried to utilize the light entering the room from the environment, which is most often un-utilized. The system varies the intensity of the light inside the room depending on the amount of light entering it.

The most innovative feature of the system is its ability to learn all the time. The system maintains a database which stores the daily usage pattern and accordingly calculates the average of it over a period of time and implements the particular pattern. This way the system is completely automated and drastically reduces human effort over a period of time.

**CHAPTER 2**

**INTELLIGENT SCHEME FOR ENERGY CONSERVATION**

**2.1 Light Intelligent Systems**

Intelligent lights (now commonly referred to as automated), can be used wherever there is a need for powerful lighting which must be capable of rapid and extreme changes of mood and effects. Moving heads would, therefore, be inappropriate in a setting which does not require strong lighting (such as a home) or where the “quality” of the light required does not vary excessively (although it may need to be very strong for a venue like a stadium). Naturally, there are exceptions to this rule, most notably the use of large numbers of moving heads for international sporting events, such as the Commonwealth Games or Olympic Games , where many thousands of separate automated fixtures are often used to light the opening and closing ceremonies. The 2008 Summer Olympics , in Beijing, had a rig of around 2,300 intelligent fixtures which is "the largest single automated lighting system ever assembled for a single event".

Usually, however, the use of intelligent lights is confined to theatre, concerts and nightclubs, where the versatility of these fixtures can be utilized to its best extent. In these applications, the uses of fixtures can be informally grouped into two categories: *active* and *passive*. Passive use of automated lighting involves utilizing their versatility to perform tasks which would otherwise require many conventional lights to accomplish. For example, six to eight moving heads can create a textured blue “night” effect on the stage floor while applying amber light to the actors during one scene - this can create a sensation of dusk or night. At the flick of a switch, the fixture can change to an animated red “fire” effect for the next scene. Attempting this transition with traditional lighting fixtures could require as many as thirty instruments. In this circumstance, the automated fixtures are not doing anything that could not be achieved using conventional fixtures, but they dramatically reduce the number of lights needed in a rig. Other features of automated fixtures, such as rotating gobos, are also possible with conventional fixtures, but are much easier to produce with intelligent fixtures.

Active use of automated lights, suggests that the luminaire is used to perform tasks which would otherwise require human involvement, or be simply impossible with conventional fixtures. For instance, a number of moving heads producing tightly-focused, pure white beams straight down onto the stage will produce a fantastic effect reminiscent of searchlights from a helicopter (especially if a smoke machine or hazer is used to make the beams visible).To recreate such an effect without intelligent lights would require at least one human operator seated directly above the stage with a follow spot, which would generally be considered to be too expensive for such a small effect.

Moving head fixtures are often divided into spot and wash lights. They vary in use and functions but many companies offer profile and wash variants of the same model of light. Profile lights generally contain features like gobos and prisms, whereas wash lights have simpler optics and a wider beam aperture resulting in wider beam angle, which may be altered by internal lenses or “frost effects”. Wash lights are more likely to have CMY colour mixing although it is common for high-end spot lights to have such features too. Spot units are generally used for their beam effect (usually through smoke or haze) and the ability to project texture, whereas wash lights tend to be used for providing a stage wash.

**Examples of a Light Intelligent System**

1. **Mercedes Intelligent Light System**

****

Fig 2.1 Mercedes Intelligent Light System

Mercedes-Benz unveils the Intelligent Light System, a new generation of adaptive car headlamps which is now entering series production. These adapt to the prevailing driving and weather conditions, thereby significantly enhancing safety. New lighting functions such as

country and motorway light modes increase the driver’s range of visibility by up to 50 metres. The Intelligent Light System also includes the active light and cornering light functions, as well as new, enhanced fog lamps which illuminate the road edges and therefore provide even better orientation when visibility is poor.

1. **Luna Eyes – The Solar Street light**

****

Fig 2.2 Luna Eyes

The Luna Eyes Solar Street light is an intelligent system, which can sense human activity and hence increases its brightness intensity level, providing convenience to pedestrians. If desired, the system can also be engineered to remain at one illumination intensity level throughout the night. These high quality eco-friendly glows reduce freight and shipping cost as they are light weighted, and compact in size. This new system is not only energy-efficient but also easy on anyone’s pocket. As far as the installation is concerned, it is extremely unproblematic. The complete lighting system is built into one compact housing unit and no complicated wiring is required. It is simply installed by clamping its brackets onto a conventional pole, without having to use special machines, vehicles or tools.

**2.2 Need for Illumination control systems**

Illumination control system finds its application in various fields. The significance and the need for such systems is going to grow with time. These systems are used for different purposes.

**Medical diagnosis**: Various areas in medicine require diagnostic equipments containing fibre light guides, which are often called illuminators for light-guide equipment. As a rule, such a device consists of a compact and efficient light source having a concentrated emitting body, an optical section that distributes the light flux and concentrates it on a small area of the input to the guide, and a heat-handling section, which combines conducting elements and forced cooling, as well as the electrical section. As the light source power and performance are improved, one has to examine the scope for operation at reduced illumination levels and the scope for choosing the levels. For example, it has been found that high-power light sources with high brightness, which are used to produce transmission by soft tissues, to the crests, and other parts of the oral cavity, are accompanied by a blinding action and tend to reduce the contrast, as has been found for example with OSM dental illuminators developed by L'vov Polytechnic Institute in collaboration with other bodies. This means that one cannot observe small details: cracks, enamel roughness, and destruction at the start of fluorosis or hypoplasia, as well as individual vessels and so on. It has also been found that one can reduce the intensity of illumination, e.g., by reducing the output light flux, to ensure the required contrast and a detailed visualization. There is a necessary illumination level for each type of detail, and good examination should be accompanied by the choice of a series of illumination levels. Also, it is not always necessary to have a high intensity of illumination in the general illumination of the oral cavity, e.g., when mouth expanders are used. Excessive intensity, particularly on prolonged illumination, causes visual fatigue and thus reduces the detail distinguish ability and the diagnostic performance.On the other hand, to produce transmission requires high intensities, which enable one in some cases to operate without the laborious, expensive, and slightly hazardous radiography, which can thus accelerate the diagnosis and also rule out secondary processes in the disease history. The need to adjust the intensity is thus evident.

**Illumination in offices:** Proper amount of light in a business office is important for health and productivity of the employees. On a sunny day, there may be more than enough light entering the office from outside to carry out the necessary tasks. Presently, the main source of light in most offices is the fluorescent lamp. Thus, it is possible to save energy by dimming the fluorescent lamp. Dimming of the fluorescent lamp is possible by changing the frequency of the sinusoidal voltage or current. Hence, there is a need of a control system to adjust the light from the fixtures based on the light entering the office from outside.Presently, the number of light fixtures in a commercial office is based entirely on the activities performed in that office. No attention is paid to the outside light entering the office from the windows. Here light control systems can be used. We can also automate the switching on and switching off of the lights in the office based on the working hours.

**Lighting in indoor gardens:** Indoor gardens are most often installed to increase the aesthetic appeal of that environment. But these plants, just like every other plant, need light for their survival. Hence the lighting in indoor spaces becomes an important aspect. The strategic placement of lights with respect to the plants is required. But the problem here is that, leaving the lights on at full intensity the whole time is obviously a waste of energy. This is where an automated system could come in handy. The intensity of the lights can be varied accordingly by making use of the light entering that particular indoor space. This way, we make efficient use of energy.

**Illuminating the class room environment**: Lighting needs to be carefully addressed in new construction and modernization projects because controlled daylight and appropriate artificial illumination are critical to the quality of student performance. What can be done to help teachers teach and help students learn? One critical area that deserves the attention of educators, administrators, designers and maintenance teams is illumination. National Clearinghouse for Education (NCEF) study report states that, in terms of lighting, there are seven independent studies indicating that classroom lighting affects student performance. These reports also document that there are optimal lighting levels for learning, that appropriate lighting improves test scores and reduces poor behavior, and that daylight fosters higher student achievement. Clearly, correct illumination is a critical component of teaching and learning Inadequate lighting controls can produce negative results, including glare, eye strain, fatigue, decreased attention span, increased body temperature and, consequently, poor student/teacher performance.

Daylight is now promoted as an energy-saver and an amenity in the classroom environment. Incorporated effectively, daylight can yield a lighter electric load and reduce heating and cooling loads to some extent, and classrooms filled with sunlight and attractive views to the outside are typically more popular with students and teachers. An intelligent illumination control system can solve this problem.

**2.3 Scheme of the light control system**

In this section we look at the components that have been used in the formulation of the scheme. This system hardware is divided into four modules as per construction: Main Board, IR Barrier Unit, Clock Generator & Power Supply Unit.

* **Main board**

|  |  |  |
| --- | --- | --- |
| **Parts List** |  |  |
| IC1 | ULN2003 | 1 Nos |
| IC2 | ADC0809 | 1 Nos |
| IC3 | 74HCT244 | 1 Nos |
| P1,P3 & P4 | 10KOhm Potentiometer | 5 Nos |
| P2 | 100KOhm Potentiometer | 1 Nos |
| D1-D3 | 1N4148 signal diodes | 3 Nos |
| LDR | 10KOhm Sensor | 1 Nos |
| Bulb | 6Volts DC Bulb | 1 Nos |
| RL1-RL3 | 5V,SPST Relays | 3 Nos |
| SW1 | SPST Switch | 1 Nos |

Table 2.1 Main Board Components

* **IR Barrier Unit**

1. **555 IR Transmitter**

**SEMICONDUCTORS**

IC1 555 1

**RESISTORS**

R1 47 Kilo-ohm 1

R2 10 Kilo-ohm 1

P1 100 Kilo-ohm Preset 1

**CAPACITORS**

C2 10μF/16v Electrolytic Capacitor 1

C1 0.001μF Ceramic Disc Capacitor 1

C3 0.01μF Ceramic Disc Capacitor 1

**MISCELLANEOUS**

LED1 Light Emitting Diode 1

IR1, IR2 Infra Red LEDs 2

B1 9V Battery 1

S1 Push-To-On switch 1

Table 2.2 IR Barrier Components

1. **555 IR Sensitive Switch**

|  |  |  |
| --- | --- | --- |
| **SEMICONDUCTORS** |  |  |
| IC1 | 555 Timer IC | 1 |
| R1 | 33 K Ohm ¼ Watt | 1 |
| R2 | 1K Ohm ¼ Watt | 1 |
| R3 | 10K Ohm ¼ Watt | 1 |
| R4 | 470 Ohm ¼ Watt | 1 |
| D1 | Red Light Emitting Diode | 1 |
| **CAPACITORS** |  |  |
| C1 & C2 | 10 µf / 25V Electrolytic | 1 |
| C2 | 0.1µF Ceramic Disc type | 1 |
| MISCELLENOUS |  |  |
| SENSOR | IR Diodes | 1 |

Table 2.3 IR Sensitive Switch

* **Clock Generator**

|  |  |  |
| --- | --- | --- |
| **EMICONDUCTORS** |  |  |
| IC1 | 555 Timer IC | 1 |
| R1 | 47 K Ohm ¼ Watt | 1 |
| R2 | 10K Ohm ¼ Watt | 1 |
| P1 | 100K Ohm Preset | 1 |
| **CAPACITORS** |  |  |
| C1 | 0.001 µf Ceramic Disc type | 1 |
| C2 | 0.01µF Ceramic Disc type | 1 |

Table 2.4 Clock Generator Components

* **Power Supply**

|  |  |  |
| --- | --- | --- |
| **SEMICONDUCTORS** |  |  |
| IC1 | 7805 Regulator IC | 1 |
| D1,D2 | 1N4007 Rectifier Diodes | 2 |
| **CAPACITORS** |  |  |
| C1 | 1000 µf/25V Electrolytic | 1 |
| C2,C3 | 0.1µF Ceramic Disc type | 1 |
| **MISCELLANEOUS** |  |  |
| X1 | 230V AC Pri,12-0-12 1Amp Sec  Transformer | 1 |

Table 2.5 Power Supply Components

**2.4 Machine Learning**

Machine learning is an integral part of Artificial Intelligence [AI]. To understand machine learning we need to first understand what Artificial Intelligence is.

The Turing Test approach, proposed by Alan Turing, suggested a test based on indistinguishability from undeniably intelligent entities-human beings. The computer passes the test if a human interrogator, after posing some written questions, cannot tell whether the written responses corne from a person or not.

.The computer would need to possess the following capabilities:

* natural language processing to enable it to communicate successfully in English.
* knowledge representation to store what it knows or hears;
* automated reasoning to use the stored information to answer questions and to draw new conclusions; machine learning to adapt to new circumstances and to detect and extrapolate patterns. Turing's test deliberately avoided direct physical interaction between the interrogator and the computer, because *physical* simulation of a person is unnecessary for intelligence. However, the so-called total Turing Test includes a video signal so that the interrogator can test the subject's perceptual abilities, as well as the opportunity for the interrogator to pass physical objects "through the hatch." To pass the total Turing Test, the computer will need
* computer vision to perceive objects, and
* robotics to manipulate objects and move about.

These six disciplines compose most of AI, and Turing deserves credit for designing a test

that remains relevant 50 years later. Yet AI researchers have devoted little effort to passing

the Turing test, believing that it is more important to study the underlying principles of intelligence than to duplicate an exemplar.

**Agents and Environments**

**ENVIRONMENT** An agent is anything that can be viewed as perceiving its environment through sensors and

**SENSOR**  acting upon that environment through actuators.

**ACTUATOR** A human agent has eyes, ears, and other organs for sensors and hands, legs, mouth, and other body parts for actuators. A robotic agent might have cameras and infrared range finders for sensors and various motors for actuators. A software agent receives keystrokes, file contents, and network packets as sensory inputs and acts on the environment by displaying on the screen, writing files, and sending network packets. We will make the general assumption that every agent can perceive its own actions (but not always the effects).

**PERCEPT** We use the term percept to refer to the agent's perceptual inputs at any given instant.

**PERCEPT SEQUENCE** agent's percept sequence is the complete history of everything the agent has ever perceived.In general, an agent's choice of action at any given instant can depend on the entire percept sequence observed to date*.* If we can specify the agent's choice of action for every possible percept sequence, then we have said more or less everything there is to say about the agent.

Mathematically speaking, we say that an agent's behavior is described by the **agent function** that maps any given percept sequence to an action. We can imagine tabulatingthe agent function that describes any given agent for most agents, this would be a very large table-infinite, in fact, unless we place a bound on the length of percept sequences we want to consider. Given an agent to experiment with, we can, in principle, construct this table by trying out all possible percept sequences and recording which actions the agent does in response. Internally*,* the agent function for an artificial agent will be implemented by an agentprogram. It is important to keep these two ideas distinct. The agent function is an abstract mathematical description~ the agent program is a concrete implementation, running on the agent architecture.

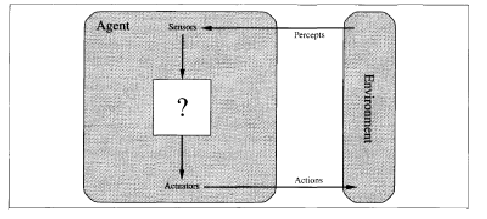
****

Fig 2.3 Agents interact with environments through sensors and actuators

**Properties of task environments**

The range of task environments that might arise in AI is obviously vast. We can, however, identify a fairly small number of dimensions along which task environments can be categorized.

* **Fully observable v/s partially observable**

If an agent's sensors give it access to the complete state of the environment at each point in time, then we say that the task environment is fully observable. A task environment is effectively fully observable if the sensors detect all aspects that are relevant to the choice of action; relevance, in turn, depends on the performance measure. Fully observable environments are convenient because the agent need not maintain any internal state to keep track of the world. An environment might be partially observable because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data-for example, a vacuum agent with only a local dirt sensor cannot tell whether there is dirt in other squares, and an automated taxi cannot see what other drivers are thinking.

* **Deterministic v/s stochastic**

If the next state of the environment is completely determined by the current state and the action executed by the agent, then we say the environment is deterministic; otherwise, it is stochastic. In principle, an agent need not worry about uncertainty in a fully observable, deterministic environment. If the environment is partially observable, however, then it could appear to be stochastic. This is particularly true if the environment is complex, making it hard to keep track of all the unobserved aspects. Thus, it is often better to think of an environment as deterministic or stochastic from the point of view of the agent. Taxi driving is clearly stochastic in this sense, because one can never predict the behavior of traffic exactly; moreover, one's tires blowout and one's engine seizes up without warning. The vacuum world as we described it is deterministic, but variations can include stochastic elements such as randomly appearing dirt and an unreliable suction mechanism. If the environment is deterministic except for the actions of other agents, we say that the environment is strategic.

* **Episodic v/s sequential**

In an episodic task environment, the agent's experience is divided into atomic episodes. Each episode consists of the agent perceiving and then performing a single action. Crucially, the next episode does not depend on the actions taken in previous episodes. In episodic environments, the choice of action in each episode depends only on the episode itself. Many classification tasks are episodic. For example, an agent that has to spot defective parts on an assembly line bases each decision on the current part, regardless of previous decisions; moreover, the current decision doesn't affect whether the next part is defective. In sequential environments, on the other hand, the current decision could affect all future decisions. Chess and taxi driving are sequential: in both cases, short-term actions can have long-term consequences. Episodic environments are much simpler than sequential environments because the agent does not need to think ahead.

* **Static v/s dynamic**

If the environment can change while an agent is deliberating, then we say the environment

is dynamic for that agent~ otherwise, it is static. Static environments are easy to deal with because the agent need not keep looking at the world while it is deciding on an action, nor need it worry about the passage of time. Dynamic environments, on the other hand, are continuously asking the agent what it wants to do, if it hasn't decided yet, that counts as deciding to do nothing. If the environment itself does not change with the passage of time but the agent's performance score does, then we say the environment is semidynamic. Taxi driving is clearly dynamic: the other cars and the taxi itself keep moving while the driving algorithm dithers about what to do next. Chess, when played with a clock, is semidynamic. Crossword puzzles are static.

* **Discrete v/s continuous**

The discrete/continuous distinction can be applied to the state of the environment, to the way time is handled, and to the percepts and actions of the agent. For example, a discrete-state environment such as a chess game has a finite number of distinct states. Chess also has a discrete set of percepts and actions. Taxi driving is a continuous state and continuous-time problem: the speed and location of the taxi and of the other vehicles sweep through a range of continuous values and do so smoothly over time. Taxi-driving actions are also continuous (steering angles, etc.). Input from digital cameras is discrete, strictly speaking, but is typically treated as representing continuously varying intensities and locations.

* **Single agent v/s multi-agent**

The distinction between single-agent and multi-agent environments may seem simple enough. For example, an agent solving a crossword puzzle by itself is clearly in a single-agent environment, whereas an agent playing chess is in a two-agent environment. There are, however, some subtle issues. First, we have described how an entity may be viewed as an agent, but we have not explained which entities must be viewed as agents. Does an agent A (the taxi driver for example) have to treat an object B (another vehicle) as an agent, or can it be treated merely as a stochastically behaving object, analogous to waves at the beach or leaves blowing in the wind? The key distinction is whether B's behavior is best described as maximizing a performance measure whose value depends on agent A's behavior. For example, in chess, the opponent entity B is trying to maximize its performance measure, which, by the rules of chess, minimizes agent A's performance measure. Thus, chess is a competitive multiagent environment. In the taxi-driving environment, on the other hand, avoiding collisions maximizes the performance measure of all agents, so it is a partially cooperative multiagent environment. It is also partially competitive because, for example, only one car can occupy a parking space. The agent-design problems arising in multiagent environments are often.

**Performance measures**

A performance measureembodies the criterion for success of an agent's behavior. When an agent is plunked down in an environment, it generates a sequence of actions according to the percepts it receives. This sequence of actions causes the environment to go through a sequence of states. If the sequence is desirable, then the agent has performed well. Obviously, there is not one fixed measure suitable for all agents. We could ask the agent for a subjective opinion of how happy it is with its own performance, but some agents would be unable to answer, and others would delude themselves. Therefore, we will insist on an objective performance measure, typically one imposed by the designer who is constructing the agent.

Consider the vacuum-cleaner agent. We might propose to measure performance by the amount of dirt cleaned up in a single eight-hour shift. With a rational agent, of course, what you ask for is what you get. A rational agent can maximize this performance measure by cleaning up the dirt, then dumping it all on the floor, then cleaning it up again, and so on. A more suitable performance measure would reward the agent for having a clean floor. For example, one point could be awarded for each clean square at each time step (perhaps with a penalty for electricity consumed and noise generated). As a general rule, it is better to design performance measures according to what One actually wants in the environment, rather than according to how One thinks the agent should behave. The selection of a performance measure is not always easy. For example, the notion of "clean floor" in the preceding paragraph is based on average cleanliness over time. Yet the same average cleanliness can be achieved by two different agents, one of which does a mediocre job all the time while the other cleans energetically but takes long breaks. Which is preferable might seem to be a fine point of janitorial science, but in fact it is a deep philosophical question with far-reaching implications. Which is better-a reckless life of highs and lows, or a safe but humdrum existence? Which is better-an economy where everyone lives in moderate poverty, or one in which some live in plenty while others are very poor? We will leave these questions as an exercise for the diligent reader.

**Types of agents**

**Simple reflex agents**

The simplest kind of agent is the simple reflex agent. These agents select actions on the basis of the current percept, ignoring the rest of the percept history. For example, the vacuum agent whose agent function is tabulated in is a simple reflex agent, because its decision is based only on the current location and on whether that contains dirt. An agent program for this agent is shown below.

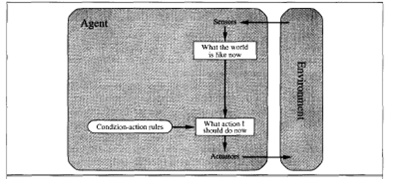


Fig 2.4 Simple Reflex Agent

Notice that the vacuum agent program is very small indeed compared to the corresponding table. The most obvious reduction comes from ignoring the percept history, which cuts down the number of possibilities from 4T to just 4. A further, small reduction comes from the fact that, when the current square is dirty, the action does not depend on the location. Imagine yourself as the driver of the automated taxi. If the car in front brakes, and its brake lights come on, then you should notice this and initiate braking. In other words, some processing is done on the visual input to establish the condition we call "The car in front is braking." Then, this triggers some established connection in the agent program to the action "initiate braking." We call such a connection a condition-action rule, written as if car-in-front-is-braking then initiate-braking. Humans also have many such connections, some of which are learned responses (as for driving) and some of which are innate reflexes (such as blinking when something approaches the eye). In the course of the book, we will see several different ways in which such connections can be learned and implemented.

The program in the Figure is specific to one particular vacuum environment. A more general and flexible approach is first to build a general-purpose interpreter for condition- action rules and then to create rule sets for specific task environments. The Figure gives the structure of this general program in schematic form, showing how the condition-action rules allow the agent to make the connection from percept to action. (Do not worry if this seems trivial; it gets more interesting shortly.) We use rectangles to denote the current internal state of the agent's decision process and ovals to represent the background information used in the process.

**Model-based reflex agents**

The most effective way to handle partial observability is for the agent to keep track of the part of the world it can't see now. That is, the agent should maintain some sort of internal

state that depends on the percept history and thereby reflects at least some of the unobserved aspects of the current state. For the braking problem, the internal state is not too extensive just the previous frame from the camera, allowing the agent to detect when two red lights at the edge of the vehicle go on or off simultaneously. For other driving tasks such as changing lanes, the agent needs to keep track of where the other cars are if it can't see them all at once. Updating this internal state information as time goes by requires two kinds of knowledge to be encoded in the agent program. First, we need some information about how the world evolves independently of the agent-for example, that an overtaking car generally will be closer behind than it was a moment ago. Second, we need some information about how the agent's own actions affect the world-for example, that when the agent turns the steering wheel clockwise, the car turns to the right or that after driving for five minutes northbound on the freeway one is usually about five miles north of where one was five minutes ago. This knowledge about "how the world works"-whether implemented in simple Boolean circuits or in complete scientific theories-is called a model of the world. An agent that uses such a model is called a model-based agent.

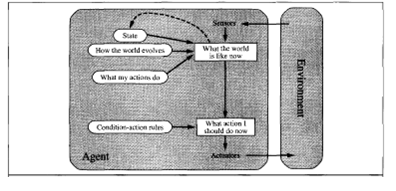
****

Fig 2.5 Model-based reflex agents

**Goal-based agents**

Knowing about the current state of the environment is not always enough to decide what to do. For example, at a road junction, the taxi can turn left, turn right, or go straight on. The correct decision depends on where the taxi is trying to get to. In other words, as well as a current state description, the agent needs some sort of goal information that describes situations that are desirable-for example, being at the passenger's destination. The agent program can combine this with information about the results of possible actions (the same information as was used to update internal state in the reflex agent) in order to choose actions

that achieve the goal. Figure shows the goal-based agent's structure. Sometimes goal-based action selection is straightforward, when goal satisfaction results immediately from a single action. Sometimes it will be more tricky, when the agent has to consider long sequences of twists and turns to find a way to achieve the goal. Search and planning are the subfields of AI devoted to finding action sequences that achieve the agent's goals. Notice that decision making of this kind is fundamentally different from the condition action rules described earlier, in that it involves consideration of the future-both "What will happen if I do such-and-suchT' and "Will that make me happy?" In the reflex agent designs, this information is not explicitly represented, because the built-in rules map directly from percepts to actions. The reflex agent brakes when it sees brake lights. A goal-based agent, in principle, could reason that if the car in front has its brake lights on, it will slow down. Given the way the world usually evolves, the only action that will achieve the goal of not hitting other cars is to brake.

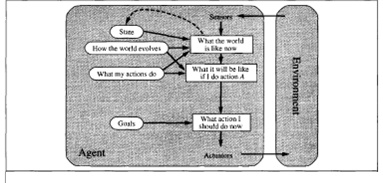
****

Fig 2.6 A model-based goal-based agent

**Utility-based agents**

Goals alone are not really enough to generate high-quality behavior in most environments. For example, there are many action sequences that will get the taxi to its destination (thereby achieving the goal) but some are quicker, safer, more reliable, or cheaper than others. Goals just provide a crude binary distinction between "happy" and "unhappy" states, whereas a more general performance measure should allow a comparison of different world states according to exactly how happy they would make the agent if they could be achieved. Because "happy" does not sound very scientific, the customary terminology is to say that if one world state is preferred to another, then it has higher utility for the agent.

A utility function maps a state (or a sequence of states) onto a real number, which describes the associated degree of happiness. A complete specification of the utility function allows rational decisions in two kinds of cases where goals are inadequate. First, when there are conflicting goals, only some of which can be achieved (for example, speed and safety), the utility function specifies the appropriate tradeoff. Second, when there are several goals

that the agent can aim for, none of which can be achieved with certainty, utility provides a way in which the likelihood of success can be weighed up against the importance of the goals.

An agent that possesses an explicit utility function therefore can make rational decisions, and it can do so via a general-purpose algorithm that does not depend on the specific utility function being maximized. In this way, the "global" definition of rationality-designating as rational those agent functions that have the highest performance-is turned into a "local" constraint on rational-agent designs that can be expressed in a simple program.

The utility-based agent structure appears in the figure.

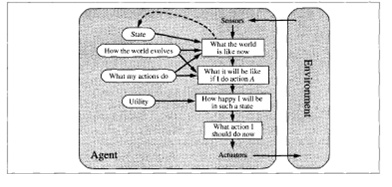
****

Fig 2.7 A model-based utility-based agent.

**Learning agents**

We have described agent programs with various methods for selecting actions. We have not, so far, explained how the agent programs come into being. In his famous early paper, Turing (1950) considers the idea of actually programming his intelligent machines by hand. He estimates how much work this might take and concludes "Some more expeditious method seems desirable." The method he proposes is to build learning machines and then to teach them. In many areas of AI, this is now the preferred method for creating state-of-the-art

systems.

Learning has another advantage, as we noted earlier: it allows the agent to operate in initially unknown environments and to become more competent than its initial knowledge

alone might allow.

A learning agent can be divided into four conceptual components, as shown in the figure . The most important distinction s between the learning element, which is responsible

for making improvements, and the performance element, which is responsible for selecting external actions. The performance element is what we have previously considered to be the entire agent: it takes in percepts and decides on actions. The learning element uses feedback from the critic on how the agent is doing and determines how the performance element should be modified to do better in the future.

The design of the learning element depends very much on the design of the performance element. When trying to design an agent that learns a certain capability, the first question is not "How am I going to get it to learn this?" but "What kind of performance element will my agent need to do this once it has learned how?" Given an agent design, learning mechanisms can be constructed to improve every part of the agent.

The critic tells the learning element how well the agent is doing with respect to a fixed performance standard. The critic is necessary because the percepts themselves provide no indication of the agent's success. For example, a chess program could receive a percept indicating that it has checkmated its opponent, but it needs a performance standard to know that this is a good thing; the percept itself does not say so. It is important that the performance standard be fixed. Conceptually, one should think of it as being outside the agent altogether, because the agent must not modify it to fit its own behavior. The last component of the learning agent is the problem generator. It is responsible for suggesting actions that will lead to new and informative experiences. The point is that if the performance element had its way, it would keep doing the actions that are best, given what it knows. But if the agent is willing to explore a little, and do some perhaps suboptimal actions in the short run, it might discover much better actions for the long run. The problem generator's job is to suggest these exploratory actions. This is what scientists do when they carry out experiments. Galileo did not think that dropping rocks from the top of a tower in Pisa was valuable in itself. He was not trying to break the rocks, nor to modify the brains of unfortunate passers-by. His aim was to modify his own brain, by identifying a better theory of the motion of objects.

To make the overall design more concrete, let us return to the automated taxi example. The performance element consists of whatever collection of knowledge and procedures the taxi has for selecting its driving actions. The taxi goes out on the road and drives, using this performance element. The critic observes the world and passes information along to the learning element. For example, after the taxi makes a quick left turn across three lanes of traffic, the critic observes the shocking language used by other drivers. From this experience, the learning element is able to formulate a rule saying this was a bad action, and the performance element is modified by installing the new rule. The problem generator might identify certain areas of behavior in need of improvement and suggest experiments, such as trying out the brakes on different road surfaces under different conditions. The learning element can make changes to any of the "knowledge" components. The simplest cases involve learning directly from the percept sequence. Observation of pairs of successive states of the environment can allow the agent to learn "How the world evolves," and observation of the results of its actions can allow the agent to learn "What my actions do." For example, if the taxi exerts a certain braking pressure when driving on a wet road, then it will soon find out how much deceleration is actually achieved. Clearly, these two learning tasks are more difficult if the environment is only partially observable.

Our system is a learning agent.

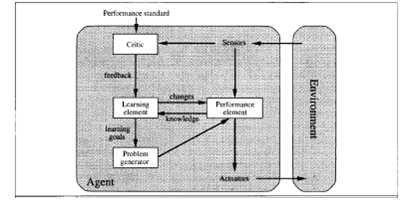
****

Fig 2.8 Learning Agent

Machine learning refers to a system capable of the autonomous acquisition and integration of knowledge. This capacity to learn from experience, analytical observation, and other means, results in a system that can improve its own speed or performance, i.e., its efficiency and/or effectiveness.

Machine-learning techniques have been used to create self-improving software for decades, but recent advances are bringing these tools into the mainstream. By Gary H. Anthes. Computerworld (February 6, 2006). "Attempts to create self-improving software date to the 1960s. But 'machine learning,' as it's often called, has remained mostly the province of academic researchers, with only a few niche applications in the commercial world, such as speech recognition and credit card fraud detection. Now, researchers say, better algorithms, more powerful computers and a few clever tricks will move it further into the mainstream.

**2.5 Procedure**

**2.5.1 Block Diagram**

ROOM LIGHT INTENSITY DETECTOR

ADC

BUFFER

USER LIGHT CONTROL UNIT

MODE SWITCH

BULB

LDR

IR BARRIER#1

IR BARRIER#2

PARALLEL PORT OF PC

AUTO LIGHT CONTROL UNIT

Fig 2.9 Block Diagram

This system allows user to set room light intensity as per his taste using knob provided for that. The user set values are fed to PC through ADC for further action. If Mode Switch is towards Auto Light Control Unit side then PC will control the light intensity of the bulb. Room Light Intensity Detector detects the current light intensity [inversely darkness] of the intended room and sends detected analog signals to ADC for digital conversion and then it is fed to PC for further processing. All signals flowing from and to hardware are through PC’s parallel port.

Two IR Barriers are used to count the people entered and exited from the room in real time.

Each Block Explanation is as follows:

**IR Barriers:** There are two Infra Red Barriers are fitted across the entrance door to detect the entering and exiting people in real time. When person crosses first barrier it is understood that he is entering the room and vice versa. Both barriers output is fed to parallel port of PC for further processing.

**Auto Light Control Unit:** This unit gets command signals from PC and accordingly sets light intensity of the bulb.

**Mode Switch:** This switch allows user to control room bulb either by PC automatically or by himself.

**Bulb:** This is fitted in the intended room and its intensity is controlled either by PC or user present in that room.

**User Light Control Unit:** This unit allows user to set the light intensity as per his desire. Its output are fed to bulb as well as ADC for conversion.

**ADC:** This Analog to Digital Converter is used to convert analog signals coming from User Light Control Unit and Room Light Intensity Detector into 8 bit digital signals.

**Buffer:** This unit acts as unit gain amplifier and isolator between hardware and PCs parallel port.

**Room Light Intensity Detector:** This unit detects the present light intensity with the help of LDR sensor and feds analogy signals to ADC for further action.

**LDR:** This Light Dependent Resistor is used to measure light intensity and varying voltages with respect to intensity are fed to Room Light Intensity Detector for processing.

**Parallel Port:** The IBM PCs parallel port is used to get signals are send commands to this system or hardware.

**2.5.2 Circuit Diagram and Description**

This system hardware is divided into four modules as per construction: Main Board, IR Barrier Unit, Clock Generator & Power Supply Unit.

User Light Control Unit: The potentiometer P3 is used to set the user defined light intensity control as per his need. This variable resistor along with Bulb generates potential divider and difference voltage produced is fed to ADC as Channel 1 input.

Auto Light Control Unit: This section receives three control signals from PC and is used to drive three low voltage & impedance relay. IC1 is a power driver and provides high voltage and current to control signals to activate relay. All relays N/C [Normally Connected] are shorted in series and one end is connected to SW1 and other to +Vcc. If control signal received is 001 then relay RL3 is activated and hence its N/O[Normally Open] pin is shorted with pole. Now P4 connected inseries is comes into action and offers preset resistance to the voltage path. Hence bulb glows with little dim intensity. Depend upon 3 bit control signal bulb intensity is varied automatically.

Room Light Intensity Detector: This section is constructed using LDR, P1 and P2. The LDR generates potential difference with P2 and is fed to Channel 0 of ADC for further action. The potentiometer P1 is connected parallel with LDR and thus controls voltage drop across it.

ADC: This IC2 is used to convert two analog signals [CH0 & CH1] into 8 bit digital signals[D0-D7] and fed to IC3 for further processing. The clock frequency for this IC is generated by Astable Multivibrator and fed at Clock input pin. The Enable Output pin is shorted with +Vcc. The Start command signal comes from PC and EOC [End Of Conversion] signal goes to PC through parallel port pins.

bulb

uln

2003

IC1

RL3

rl2

rl1

D3 C3

D2 C2

**D1 C1**

P4

P4

P4

Sw1

P3

+Vcc

LDR P1

P2

adc

0809

IC2

CH0

ch1

Vcc  Eo

EOC S1

START C4

74HCT244

IC3

D0

D1

D2

D3

D4  
D5

D6

D7

D0

D1

D2

D3

D4  
D5

D6

D7

Gnd

+Vcc

+Vcc

CLK

AMV

Fig 2.10 Circuit Diagram of the Main Board

**555 IR Transmitter**

This circuit transmits the Infrared packets to the IR receiver with different frequency range. The frequency can be set by changing the resistor R & capacitor C values [RC time constant]. Shown typical circuit transmits IR packets of 36 kilo hertz frequency.

**Timer:** An electronic circuit that generates square waves using positive feedback is known as a Multivibrator. This switching circuit is basically a two stage amplifier and operates in two states (ON and OFF) controlled by external circuit conditions. There are 3 types of Multivibrators: Astable or Free Running Multivibrator, Monostable or One Shot Multivibrator and Bistable or Flip-flop Multivibrator.

Digital circuits often require a source of accurately defined pulses. The requirement is generally for a single pulse of given duration (i.e. a ‘one shot’) or for a continuous train of pulses of given frequency and duty cycle. Rather than attempt to produce an arrangement of standard logic gates to meet these requirements, it is usually simpler and more cost-effective to make use of one of the range of versatile integrated circuits known collectively as ‘timers’. The greater level of accuracy and stability with long Monostable periods is possible only with timer IC. The 555 timer is a neat mixture of analogue and circuitry but its applications are virtually limitless in the world of digital pulse generation. These devices can usually be configured for either Monostable or Astable operation and require only a few external components in order to determine their operational parameters.By combining Monostable Multivibrator and IR diodes, one can build an IR Transmitter.

**Internal Arrangement of 555 Timer IC**

The timer comprises two operational amplifiers (used as comparators) together with an RS bistable element. In addition, an inverting output buffer is incorporated so that a considerable current can be sourced or sunk to/from a load. A single transistor switch, TR1, is also provided as a means of rapidly discharging the external timing capacitor.

The standard 555 timer is housed in an 8-pin DIL package and operates from supply rail voltages of between 4.5V and 15V. This encompasses the normal range for TTL devices and thus the device is ideally suited for use in conjunction with TTL circuitry.

RESET

OUTPUT

TRIGGER

VCC

555

8

7

6

5

2

3

1

4

DISCHARGE

THRESHOLD

GROUND

CONTROL

Fig 2.11 Pin Out Diagram Of Timer IC 555

**IC1**

4

8

7

3

1

6

2

5

S1

R1

P1

R2

B1

C1

C3

Reset

Vcc

C2

O/P

LED1

IR1

IR2

Fig 2.12 Circuit Diagram of IR Transmitter

An electronic circuit that generates square waves using positive feedback is known as a Multivibrator. This switching circuit is basically a two stage amplifier and operates in two states (ON and OFF) controlled by external circuit conditions. There are three types of Multivibrator: Astable or Free Running Multivibrator, Monostable or One Shot Multivibrator and Bistable or Flip-flop Multivibrator.

An oscillator circuit which generates square wave of its own (i.e. without external triggering) is known as Astable or Free Running Multivibrator. The outputted square pulse is not stable in nature. It switches back and forth from one state to the other. And the switching time is determined by the external components (i.e. RC constant). These pulse trains are used to ON/OFF the connected pair of Infrared LEDs.

The circuit diagram shows how the timer IC 555 can be used as an Astable pulse generator. In this mode the circuit provides very constant output frequency of 36 kilo hertz. As the circuit is self-triggering, trigger pulse input pin-2 is grounded through capacitor C1. When the circuit is first put ON, the capacitor C1 is uncharged and the trigger input is low and that switching transistor TR1 (at pin-7) is in the non-conducting state. Thus the output (at pin-3) is high. The capacitor C1 will begin to charge toward +Vcc with current supplied by means of the series resistors R1,P1 and R2.

When the voltage at the ‘threshold’ input (at pin-6) exceeds ⅔ of Vcc, the output of the upper comparator will change state and the Bistable will be reset, making the Ō output go ’HIGH’ and turning TR1 ‘ON’ in the process. Due to the inverting action of the buffer, the final ‘output’ (at pin-3) will then go ‘LOW’.

The capacitor C1 will now discharge, with current flowing through R2 & P1 into the collector of switching transistor TR1 (at pin-7). At a certain point, the voltage appearing at the ‘trigger’ input (pin-2) will have fallen back to one third of the supply voltage at which point the lower comparator will change state and return the Bistable to its original set condition. The Q ‘output’ of the Bistable then goes low, TR1 switches ‘off’, and the final ‘output’ (pin-3) goes high. Thereafter the entire cycle is repeated indefinitely.

The essential characteristics of this waveform are:

|  |  |
| --- | --- |
| Time for which output is ‘high’: | Ton=0.693(R1+R2+P1) C |
| Time for which output is ‘low’: | Toff=0.693(R2+P1) C |
| Period of output: | T=Ton+Toff=0.693(R1+P1+2R2) C |
| Pulse Repetitive Frequency of output: | p.r.f. = 1.44 / (R1+P1+2R2) C |
| Pulse Period: | T = 1/ p.r.f |

Where T is in seconds, C is in farads, and R1 & R2 are in Ohms.

The outputted pulses are given to pair of Infrared Light Emitting Diodes IR1 & IR2, which blinks at 36 KHZ frequency. LED1 indicates the presence of pulses at output pin-3.

Note: It should be noted that the mark to space ratio produced by a 555 timer can never be less than unity (i.e. 1:1). However, by making R2 very much large than R1 the timer can be made to produce a reasonably symmetrical square wave.

**555 IR Sensitive Switch**

The circuit diagram shows how the timer IC 555 can be used as a Rising IR Level Switch. In Monostable pulse generator mode, pin 4 is connected to pin 8 and that to +Vcc. The threshold pin 6 and the discharge pin 7 are connected together to +Vcc by a resistance R3. The control pin 5 is connected to ground via capacitor C2. The trigger input pin 2 is connected to +Vcc using a pull-up resistor R1.Here the IR sensor Diode, R2 & C1 gives the triggering pulse needed for Multivibrator.

The current through IR diode will depend upon the amount of incident light. In total NO IR Rays the reverse current flowing through IR diode will be very small. When the sensor IR diode is not illuminated by a light source the capacitor C2 is uncharged and the trigger input is low and that switching transistor TR1 (at pin-7) is in the non-conducting state. Thus the output (at pin-3) is high. The capacitor C1 will begin to charge toward +Vcc with current supplied by means of the series resistors R1 and R2.

C1

4 8

3

2

D1

GND

R1

R2

R4 470Ω

R3

C2

C3

SENSOR

Output

+Vcc

6

7

1 5

**555**

Fig 2.13 555 IR Sensitive Switch

When the light source is focussed on the IR diode, the reverse current flowing through IR diode increases markedly. Thus Monostable timing period is initiated by a falling edge (i.e. ‘High’ to ‘Low’ transition) applied to the trigger input (at pin 2). When such an edge is received and the ‘trigger’ input voltage falls below ⅓ of Vcc, the output of the lower comparator goes ‘high’ and the Bistable is placed in the ‘set’ state. The Q output of the Bistable then goes low, switching transistor TR1 is placed in the ‘OFF’ (non-conducting) state and the final ‘output’ (at pin-3) goes High. The circuit can be readily adapted to drive a load with operating current less than about 150mA. So, the indicator LED (D1) goes ‘ON’ stating the load is in ON position. The output is latched in the same condition until the light source is not blocked or stopped.

When the illuminating light source is blocked or stopped the internal resistance of the IR diode increases dramatically. So there is a decay of reverse current flowing through it is observed. This Low-to-High transition exceeds the voltage at ‘threshold’ input (at pin-6) to ⅔ of Vcc. Thus output of the upper comparator will change state and the Bistable will be reset, making the Ō output go ’HIGH’ and turning switching transistor TR1 ‘ON’ in the process. Due to the inverting action of the buffer, the final ‘output’ (at pin-3) will then go ‘LOW’. That means the indicator LED (D2) goes ‘OFF’ stating the load is in OFF position.

**Clock Generator**

The clock pulse required for the ADC0809 was given by an IC555 in astable mode. The clock frequency that can be supplied to ADC0809 is between 200-640KHz. We designed our clock for frequency of 227.27 KHz.

Introduction: Digital circuits often require a source of accurately defined pulses. The requirement is generally for a single pulse of given duration (i.e. a ‘one shot’) or for a continuous train of pulses of given frequency and duty cycle. Rather than attempt to produce an arrangement of standard logic gates to meet these requirements, it is usually simpler and more cost-effective to make use of one of the range of versatile integrated circuits known collectively as ‘timers’. The greater level of accuracy and stability with long Monostable periods is possible only with timer IC. The 555 timer is a neat mixture of analogue and circuitry but its applications are virtually limitless in the world of digital pulse generation. These devices can usually be configured for wither monostable or astable operation and require only a few external components in order to determine their operational parameters.

INTERNAL ARRANGEMENT OF 555 TIMER IC

The timer comprises two operational amplifiers (used as comparators) together with an RS bistable element. In addition, an inverting output buffer is incorporated so that a considerable current can be sourced or sunk to/from a load. A single transistor switch, TR1, is also provided as a means of rapidly discharging the external timing capacitor.

The standard 555 timer is housed in an 8-pin DIL package and operates from supply rail voltages of between 4.5V and 15V. This encompasses the normal range for TTL devices and thus the device is ideally suited for use in conjunction with TTL circuitry.

R2

P1 100K

C1

4

8

7

3

1

6

2

5

+Vcc

Output

C2

R1 47K

**555**

Gnd

Fig 2.14 Astable Multivibrator using 555 Timer IC

An electronic circuit that generates square waves using positive feedback is known as a Multivibrator. This switching circuit is basically a two stage amplifier and operates in two states (ON and OFF) controlled by external circuit conditions. There are three types of Multivibrators: Astable or Free Running Multivibrator, Monostable or One Shot Multivibrator and Bistable or Flip-flop Multivibrator.

An oscillator circuit which generates square wave of its own (i.e. without external triggering) is known as Astable or Free Running Multivibrator. The outputted square pulse is not stable in nature. It switches back and forth from one state to the other. And the switching time is determined by the external components (i.e. RC constant). These pulse trains are used to ON/OFF or trigger the connected external circuits. The normal 555 IC Astable Multivibrator can be used readily to drive a relay (operating current must be less than 150mA).

The circuit diagram shows how the timer IC 555 can be used as an astable pulse generator. In this mode the circuit provides very constant output frequency. As the circuit is self-triggering, trigger pulse input pin-2 is grounded through capacitor C1. When the circuit is first put ON, the capacitor C1 is uncharged and the trigger input is low and that switching transistor TR1 (at pin-7) is in the non-conducting state. Thus the output (at pin-3) is high. The capacitor C1 will begin to charge toward +Vcc with current supplied by means of the series resistors R1,P1 and R2.

When the voltage at the ‘threshold’ input (at pin-6) exceeds ⅔ of Vcc, the output of the upper comparator will change state and the bistable will be reset, making the Ō output go ’HIGH’ and turning TR1 ‘ON’ in the process. Due to the inverting action of the buffer, the final ‘output’ (at pin-3) will then go ‘LOW’.

The capacitor C1 will now discharge, with current flowing through R2 & P1 into the collector of switching transistor TR1 (at pin-7). At a certain point, the voltage appearing at the ‘trigger’ input (pin-2) will have fallen back to one third of the supply voltage at which point the lower comparator will change state and return the bistable to its original set condition. The Q ‘output’ of the bistable then goes low, TR1 switches ‘off’, and the final ‘output’ (pin-3) goes high. Thereafter the entire cycle is repeated indefinitely.

The essential characteristics of this waveform are:

|  |  |
| --- | --- |
| Time for which output is ‘high’: | Ton=0.693(R1+R2+P1) C |
| Time for which output is ‘low’: | Toff=0.693(R2+P1) C |
| Period of output: | T=Ton+Toff=0.693(R1+P1+2R2) C |
| Pulse Repetitive Frequency of output: | p.r.f. = 1.44 / (R1+P1+2R2) C |
| Pulse Period: | T = 1/ p.r.f |

Where T is in seconds, C is in farads, and R1 & R2 are in Ohms.

**Power Supply Unit**

The power supply, unsung hero of every electronic circuit, plays very important role in smooth running of the connected circuit. The main object of this ‘power supply’ is, as the name itself implies, to deliver the required amount of stabilized and pure power to the circuit. Every typical power supply contains the following sections:

1. Step-down Transformer: The conventional supply, which is generally available to the user, is 230V AC. It is necessary to step down the mains supply to the desired level. This is achieved by using suitably rated step-down transformer. While designing the power supply, it is necessary to go for little higher rating transformer than the required one. The reason for this is, for proper working of the regulator IC (say KIA 7805) it needs at least 2.5V more than the expected output voltage

2. Rectifier stage: Then the step-downed Alternating Current is converted into Direct Current. This rectification is achieved by using passive components such as diodes. If the power supply is designed for low voltage/current drawing loads/circuits (say +5V), it is sufficient to employ full-wave rectifier with centre-tap transformer as a power source. While choosing the diodes the PIV rating is taken into consideration.

3. Filter stage: But this rectified output contains some percentage of superimposed a.c. ripples. So to filter these a.c. components filter stage is built around the rectifier stage. The cheap, reliable, simple and effective filtering for low current drawing loads (say upto 50 mA) is done by using shunt capacitors. This electrolytic capacitor has polarities, take care while connecting the circuit.

4. Voltage Regulation: The filtered d.c. output is not stable. It varies in accordance with the fluctuations in mains supply or varying load current. This variation of load current is observed due to voltage drop in transformer windings, rectifier and filter circuit. These variations in d.c. output voltage may cause inaccurate or erratic operation or even malfunctioning of many electronic circuits. For example, the circuit boards which are implanted by CMOS or TTL ICs.

The stabilization of d.c. output is achieved by using the three terminal voltage regulator IC. This regulator IC comes in two flavors: 78xx for positive voltage output and 79xx for negative voltage output. For example 7805 gives +5V output and 7905 gives -5V stabilized output. These regulator ICs have in-built short-circuit protection and auto-thermal cutout provisions. If the load current is very high the IC needs ‘heat sink’ to dissipate the internally generated power.

B

230 AC

# X1

0 V

IC 1

C1

D2 221

C2

C3

D1 111

A

O

C



D

Ground

E

+5V

Fig 2.15 Power Supply

A d.c. power supply which maintains the output voltage constant irrespective of a.c. mains fluctuations or load variations is known as regulated d.c. power supply. It is also referred as full-wave regulated power supply as it uses two diodes in full wave fashion with centre tap transformer.

1.Step-down Transformer : The transformer rating is 230V AC at Primary and 12-0-12V, 1Ampers across secondary winding. This transformer has a capability to deliver a current of 1Ampere, which is more than enough to drive any electronic circuit or varying load. The 12VAC appearing across the secondary is the RMS value of the waveform and the peak value would be 12 x 1.414 = 16.8 volts. This value limits our choice of rectifier diode as 1N4007, which is having PIV rating more than 16Volts.

2. Rectifier Stage : The two diodes D1 & D2 are connected across the secondary winding of the transformer as a full-wave rectifier. During the positive half-cycle of secondary voltage, the end A of the secondary winding becomes positive and end B negative. This makes the diode D1 forward biased and diode D2 reverse biased. Therefore diode D1 conducts while diode D2 does not. During the negative half-cycle, end A of the secondary winding becomes negative and end B positive. Therefore diode D2 conducts while diode D1 does not. Note that current across the centre tap terminal is in the same direction for both half-cycles of input a.c. voltage. Therefore, pulsating d.c. is obtained at point ‘C’ with respect to Ground.

3.Filter Stage : Here Capacitor C1 is used for filtering purpose and connected across the rectifier output. It filters the a.c. components present in the rectified d.c. and gives steady d.c. voltage. As the rectifier voltage increases, it charges the capacitor and also supplies current to the load. When capacitor is charged to the peak value of the rectifier voltage, rectifier voltage starts to decrease. As the next voltage peak immediately recharges the capacitor, the discharge period is of very small duration. Due to this continuous charge-discharge-recharge cycle very little ripple is observed in the filtered output. Moreover, output voltage is higher as it remains substantially near the peak value of rectifier output voltage. This phenomenon is also explained in other form as: the shunt capacitor offers a low reactance path to the a.c. components of current and open circuit to d.c. component. During positive half cycle the capacitor stores energy in the form of electrostatic field. During negative half cycle, the filter capacitor releases stored energy to the load.

4.Voltage Regulation Stage : Across the point ‘D’ and Ground there is rectified and filtered d.c. In the present circuit KIA 7805 voltage regulator IC is used to get +5V regulated d.c. output. In the three terminals, pin 1 is input i.e., rectified & filtered d.c. is connected to this pin. Pin 2 is common pin and is grounded. The pin 3 gives the stabilized d.c. output to the load. The circuit shows two more decoupling capacitors C2 & C3, which provides ground path to the high frequency noise signals. Across the point ‘E’ and ground +5V stabilized or regulated d.c output is measured, which can be connected to the required circuit.

Note: While connecting the diodes and electrolytic capacitors the polarities must be taken into consideration. The transformer’s primary winding deals with 230V mains, care should be taken with it.

**2.5.3 Parallel Port Connection**

A parallel port is a type of interface found on [computers](http://en.wikipedia.org/wiki/Computers) ([personal](http://en.wikipedia.org/wiki/Personal_computer) and otherwise) for connecting various peripherals. In [computing](http://en.wikipedia.org/wiki/Computing), a parallel port is a [parallel communication](http://en.wikipedia.org/wiki/Parallel_communication) physical interface. The [IEEE 1284](http://en.wikipedia.org/wiki/IEEE_1284) standard defines the bi-directional version of the port. This transmits particular amount of bits in parallel at the same time. This is opposite to [serial](http://en.wikipedia.org/wiki/Serial_cable) transition where one bit will be transmitted at a time.

Parallel port is a simple and inexpensive tool for building computer controlled devices and projects. The simplicity and ease of programming makes parallel port popular in electronics hobbyist world. The parallel port is often used in Computer controlled robots, Atmel/PIC programmers, home automation, ...etc...

The Parallel Port is the most commonly used port for interfacing home-made projects. This port will allow the input of up to 9 bits or the output of 12 bits at any one given time, thus requiring minimal external circuitry to implement many simpler tasks. The port is composed of 4 control lines, 5 status lines and 8 data lines. It's found commonly on the back of your PC as a D-Type 25 Pin female connector. There may also be a D-Type 25 pin male connector. This will be a serial RS-232 port and thus, is a totally incompatible port.

           Everybody knows what is parallel port, where it can be found, and for what it is being used. the primary use of parallel port is to connect printers to computer and is specifically designed for this purpose. Thus it is often called as printer Port or Centronics port (this name came from a popular printer manufacturing company 'Centronics' who devised some standards for parallel port). You can see the parallel port connector in the rear panel of your PC. It is a 25 pin female (DB25) connector (to which printer is connected). On almost all the PCs only one parallel port is present, but you can add more by buying and inserting ISA/PCI parallel port cards.   The five modes of data transfer for parallel port. They are,

* Compatibility Mode
* Nibble Mode
* Byte Mode
* EPP
* ECP

           The programs, circuits and other information found in this tutorial are compatible to almost all types of parallel ports and can be used without any problems .

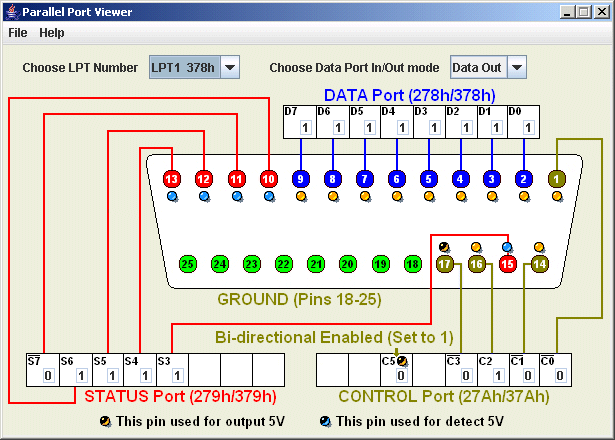


Fig 2.16 Parallel Port Pins

The lines in DB25 connector are divided in to three groups, they are

* Data lines (data bus)
* Control lines
* Status lines

           As the name refers , data is transferred over data lines , Control lines are used to control the peripheral and of course , the peripheral returns status signals back computer through Status lines. These lines are connected to Data, Control and Status registers internally.

Almost all programming languages allow programmers to access parallel port using some library functions. For example, Borland C is providing "Inportb" and "Outportb" functions to read or write IO mapped peripherals. But the examples provided here in this tutorial is written VC++ and can be easily ported to other compilers like Borland C and Turbo C. Visual Basic does not have any functions or support to access parallel port directly, but it is possible to add such capabilities to your VB application by writing a dll in VC++ and calling its exported functions from VB. VC++ provides two functions to access IO mapped peripherals, '\_inp' for reading and '\_outp' for writing. These functions are declared in "conio.h".

The Parallel Port has three commonly used base addresses. These are listed in the table below. The 3BCh base address was originally introduced used for Parallel Ports on early Video Cards. This address then disappeared for a while, when Parallel Ports were later removed from Video Cards. They has now reappeared as an option for Parallel Ports integrated onto motherboards, upon which their configuration can be changed using BIOS.

LPT1 is normally assigned base address 378h, while LPT2 is assigned 278h. However this may not always be the case as explained later. 378h & 278h have always been commonly used for Parallel Ports. The lower case h denotes that it is in hexadecimal. These addresses may change from machine to machine.

|  |  |
| --- | --- |
| Address | Notes: |
| 3BCh - 3BFh | Used for Parallel Ports which were incorporated on to Video Cards - Doesn't support ECP addresses |
| 378h - 37Fh | Usual Address For LPT 1 |
| 278h - 27Fh | Usual Address For LPT 2 |

Table 2.6 Port Addresses



Fig 2.17 Parallel Port

**2.5.4 Working of the system**

The working of the whole system can be summarized in a block diagram shown below:

**Counting Sensor**

**Relay Circuit**

**Intensity Sensor**

****

**Intensity Sensor**

CPU

Fig 2.18 Working of the System

* The values obtained from the intensity sensor and the counting sensor is sent to the CPU.
* The system makes use of the various algorithms and sends an output value and the required light intensity is implemented with the help of relays.
* There is a knob present for the user to make changes in the intensity of the light if he’s not happy with the output of the system.

**CHAPTER 3**

**HARDWARE AND SOFTWARE REQUIREMENTS**

**3.1 Hardware requirements**

The basic hardware required for the development of the artifact is:-

1. Computer system.

The configuration required of the computer system being used for the development of the artifact is as follows:-

* Processor - 600 MHz
* RAM - 256 MB
* Hard disk - 1 GB
* Video - 800 x 600, 256 colors , High Color 16-bit
* Presence of the parallel port in the system.

The other hardware components required of the development of the circuit of the artifact are:-

1. Light Measuring Sensor --->One to measure the intensity of the ambient light entering the room and a second one to measure the intensity of light inside the room.
2. Infra-Red Sensor---> Used in order to keep the count of the people entering and leaving the room.
3. Db25---> It is the cable required to interface with the parallel port of the computer.
4. Analog to digital converter---> A 8-bit converter is required to conver the analog values of the sensor to computer understandable form.
5. Relay---> It is the component required in order to connect the bulb or the artificial light present inside the room to the rest of the circuit.
6. Led’s ---> Used to demonstrate the working of the system. It used as the artificial light present inside the room.

**3.2 Software requirements**

1. Operating system – Windows XP
2. JAVA ( J2SE 6 )
3. JDBC
4. Eclipse Platforms
5. MYSQL
6. Microsoft Visual Studio C++

**CHAPTER 4**

**IMPLEMENTATION OF THE SCHEME AND RESULTS**

**4.1 Opening the Database**

First thing to be done before actually using the database i.e. reading to a database or writing to it, is to open the database and then interface the database with the programming language in use.

The database or backhand which is being used for the development of the system is MYSQL. The various steps required in order to open the database are:-

Step1->Start the MYSQL using XAMPP.

Step2-> Go to start menu and select command prompt.

Step3->Change the directory to the root directory by using cd.. command.

Step4->Type mysql -u root –p to open MYSQL.

Step5->After executing the above command you will be asked to enter the password.

Step6-> Enter your SQL password if any.

Step7->Change the the database to your required one.

Step8->Type use <database name> in order to change to required database.

Step9->After doing this create the required tables and insert the values into those tables.

Once the Database is created and the values are inserted into the tables of the database the values can be retrieved, modified and deleted as per the requirements of the user or the system being developed by the user.

**4.2 Connecting the Database**

To Connect to the database using java the Following steps must be followed:-

Step1->Create an object of type connection in order to connect to database.

Step2->Create an object of type statement to execute the query.

Step3->Create an object of type result set to store the results fetched by query.

Step4->Create two string variables usr and pass to store the username and password of the MYSQL.

Step5->Create string variable url which is used to store the url of the database.

Step6->Use Class.forName () function to initialize the driver to be used to interface with database.

Step7->Use DriverManager.getConnection() function in order to get the Connection string required to access the database.

Step8-> Use the above connection string to create the query and execute thee query.

Step9->Use the resultSet object to access the result generated by the execution of the query.

Below shown is the entire java code to open a connection to the database , create a query, execute a query and then print the result produced by the execution of the query:-

import java.sql.\*;

public class test1 {

public static void main (String[] args) throws Exception{

Connection db;

Statement st;

ResultSet rs = null;

String url = "jdbc:mysql://localhost:3306/temp";

String query = null;

String usr = "root";

String pass = "";

try {

Class.forName("com.mysql.jdbc.Driver");

db = DriverManager.getConnection(url,usr,pass);

st = db.createStatement();

for (int i = 12; i<20 ; i++){

query = "insert into temp values (" + i + ");";

int n = st.executeUpdate(query);

}

query = "select \* from temp;";

rs = st.executeQuery(query);

while (rs.next())

{

System.out.println(rs.getString(1));

}

}

catch (Exception e1){

System.err.println(e1);

}

}

}

**4.3 Database Design**

**LIGHT\_TABLE (Data Storage) :-**

Light\_table stores the feedback values for each hour of the day. Feedback values are the measure of the total intensity in the room at the given time. This includes the natural light entering the room from outside and the artificial light of the lamps. This value is used to calculate the usage pattern of the user over a period of time.

**Name Constraint Type**

Sno NOT NULL int

AUTO\_INCREMENT

Dt date

Weekday varchar(10)

fd1 int

fd2 int

… int

… int

… int

… int

fd24 int

**ML\_TABLE (Machine Learning) :-**

This table stores the learned values for 24 hours of the day for each weekday. This table has 7 rows and 25 columns. The learned values for each hour are calculated based on feedback values for the same hour of day for the same weekday for the last 15 weeks.

**Name Constraint Type**

Weekday NOT NULL varchar(10)

lv1 int

lv2 int

lv3 int

… int

… int

… int

… int

… int

… int

… int

lv24 int

**4.4 The Overall Implementation of the Scheme**

The implementation of the scheme is carried out with the help of five different algorithms which perform various functionalities in order to accomplish the goal for which the system is designed. The five different algorithms are implemented as the five different functions of the main program. The five functions are as follows:-

1) The Control algorithm named as control\_algo.

2) The Wait algorithm named as wait\_algo.

3) The learn algorithm named as learn\_algo.

4) The sum algorithm named as sum\_algo.

5) The average algorithm named as avg\_algo.

The execution of the program begins with the main function which in turn calls the control\_algo function and the flow of control continues from there onwards. The various functionalities carried out by the different functions are as described below:-

**4.4.1 The control\_algo() Function**

The control\_algo is the function from which the execution starts. The function is called by the main function of the program. The function runs a infinite while loop to check the number of people currently present inside the room with the help of the global variable count and measures the intensity of the sunlight entering the room which in turn is stored in another global variable intensity\_input. The function first checks if count is equal to zero and if so then the light inside the room is switched off automatically and a call is made to wait\_algo function to wait for a time period of 4 second and then again checks for the count value and the same procedure is carried out again.

The next condition is if the value of count is greater than zero if so then learned intensity value of that particular day and hour is retrieved from the ml\_table (which is a table used in order to store the machine learned values) and then the difference of learned value of intensity and the input intensity is calculated and that value is set as the current intensity of the artificial light. This process is carried out for each and every hour of the day.

The algorithm is as follows:

ControlAlgo (Intensity\_ip, Count\_ip, Op\_port)

Intensity\_ip : Intensity of ambient light.

Count\_ip : Count of number of people inside the room.

Op\_port : Port to change the intensity of the room light.

BEGIN

Loop1: If (Count\_ip ==0) then //Checks if the number of people in a room are 0.

Op\_port =0;

WaitAlgo(Intensity\_ip);

Else

//gets the learned intensity from the database.

X = select learned from ML\_TABLE where weekday = today

//Calculates the value of intensity of the room light based on learned value and current ambient light intensity.

Op\_port = X - intensity\_ip;

WaitAlgo(Intensity\_ip);

END

**4.4.2 The wait\_algo() Function**

The wait\_algo() is the function that is being called by the control\_algo() function. The main task of this particular function is to control the execution order the remaining 3 functions. In order to achieve this, the function first retrieves the current hour, minute and second of the system and then accordingly makes the call to desired functions depending on the specified conditions. The various time durations in which the functions are called by the wait are:-

1. The sum\_algo is called once in every minute.
2. The avg\_algo is called once in every hour.
3. The learn\_algo is called once every day that is when hour =0 minute =0 and second=0.

The other thing that is done by this function is that it checks whether then change in the intensity is more than 10% as compared to the old intensity if this condition is satisfied than only the intensity of the light present inside the room is varied else it is left as it is.

The wait\_algorithm is as follows:

WaitAlgo(old\_intensity, count\_ip, Intensity\_ip)

old\_intensity : Intensity reading passed from ControlAlgo

count\_ip : Count value from the input port

Intensity\_ip : Intensity reading from the input port

BEGIN

new\_intensity = Intensity\_ip;

if (count != 0)

{

If (( new\_intensity\_ip >= old\_intensity \* 110/100) || new\_intensity\_ip <= old\_intensity \* 90/100) then

//System reevaluates if the intensity of ambient light changes by more than 10%.

Return();

}

else

wait(30\*1000);

END

**4.4.3 The sum\_algo() Function**

The sum\_algo() function is called by the wait\_algo() function after every minute. The task of this function is to calculate the sum feedback of the intensity of the light and store it in a global variable named sum\_feedback. The value stored in this variable is calculated as the sum of input intensity and the knob input.

The values stored in these variables are:-

1. Input\_intensity---> It stores the intensity of the sunlight that is being measured with the help of sensor. This intensity is basically the intensity of the sunlight entering the room.
2. Knob\_input---> It store the changed intensity of the light present inside the room. This is changed by the user himself.

The algorithm is as follows:

SumAlgo(Sum\_Intensity, Current\_Intensity, Sum\_Feedback, Current\_feeback)

//Executes every minute.

Sum\_Intensity : Sum of the intensity readings.

Current\_Intensity : As passed by the sensor.

Sum\_Feedback : Sum of the feedback values.

Current\_feeback : Current feedback.

INITIALIZE:

Sum\_Intensity = 0

Sum\_Feedback = 0

BEGIN

Sum\_Intensity = Sum\_Intensity + Current\_Intensity;

Sum\_Feedback = Sum\_Feedback + Current\_Feedback;

END

**4.4.4 The avg\_algo() Function**

The avg\_algo function is being called by the wait-algo() function after every hour. The basic work of this function is to calculate the average intensity of the light for every single hour of the day. This function calculates the average intensity simply by dividing the sum\_feedback intensity calculated by sum\_algo() function by 60. Once this average feedback value is calculated by this function this average value is stored in the feedback table present in the database. This value is calculated is appropriate column of the table which is named by that particular hour of that particular day.

The algorithm is as follows:

AverageAlgo (Average\_Feedback, Sum\_Feedback)

//Executes every hour

Sum\_Feedback : Variable that contains the sum of feedback values.

Average\_Feedback : Variable to store the average Feedback.

BEGIN

//Calculate the average feedback.

Average\_Feedback = Sum\_Feedback / 60;

//Insert the average value into data\_table of the database in the appropriate field.

UPDATE data\_table SET feedback = Average\_Feedback WHERE date = CurrentDate();

END

**4.4.5 The learn\_algo() Function**

The learn\_algo() function is the most important function and it is the one with the help of which the machine learning concept is implemented in the system. This function inturn is again called by the wait\_algo() function after every day or we can say that after every 24 hours since when the system is started. The functionality carried by this particular function is that it calculates the intensity of particular hour and particular day of the week as the average of the last 15 days where the day is same as current day and the hour of that day is same as the current hour of the day. After calculating the intensity for particular day and hour it stores that value in the ml\_table(which is a table used in order to store the machine learned values) for this particular hour of the day and hence this stored value is fetched and used in order to implement the machine learning concept of the system.

The algorithm is as follows:

LearnAlgo( Learned\_Intensity, Sum\_Learned\_Intensity)

// Executes once a day.

Learned\_Intensity [24] : The intensity values learned for a day.

Sum\_Learned\_Intensity : Contains the sum of intensities of the last 15 weeks.

BEGIN

For i = 1 to 24 do

For j=1 to 15

//Find out the sum of the feedback intensity from the data\_table for the current weekday.

Sum\_Learned\_Intensity = Sum\_Learned\_Intensity + SELECT feedback(i) FROM data\_table WHERE date = CurrentDate() – (7\*j);

END j

//Update the table ML\_TABLE for all the hours of the day.

UPDATE ML\_TABLE SET Learned(i) = Sum\_Learned\_Intensity / 15;

END i

END

**4.4.6 Code**

The full program code to implement the above proposed concept is as follows:-

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.text.SimpleDateFormat;

import java.util.\*;

// main class of the program

public class demo {

public static float current\_feedback=0, sum\_feedback=0, old\_intensity=0, intensity\_input=0, knob\_feedback=0;

public static String query = null;

public static Connection db;

public static Statement st;

public static ResultSet rs = null;

public static String url = "jdbc:mysql://localhost:3306/light";

public static String usr = "root";

public static String pass = "";

public static int count = 0;

public static int range[]={14,18,16,20,14,18,10,15,0,5,10,15};

public demo(){

// try block to open the connection with database

try{

Class.forName("com.mysql.jdbc.Driver");

db = DriverManager.getConnection(url,usr,pass);

st = db.createStatement();

}

// catch block to catch exception if generated by while opening a connection

catch (SQLException e1){

System.err.println(e1);

}

catch (ClassNotFoundException e2){

System.err.println(e2);

}

}

// random function to randomly increase and decrease the count of people

public static void random\_count(){

Random rn1 = new Random(); // in bulid random function;

int r1 = rn1.nextInt(2);

if (r1 == 1){

Random rn =new Random();

int r =rn.nextInt(10);

if(r<=5){

count++; // count increased if random no. is less than equal to 5;

}

else{

count--; // count decreased if random no. is greater than 5;

if(count < 0){

count = 0; // count is made 0 if count goes negative;

}

r= r-5;

}

}

}

//random function which randomly generates the value for intensity measured by sensor

public static int random\_intensity(){

int temp;

String dateFormat = "H"; // command to fetch the current system hour

SimpleDateFormat sdf2 = new SimpleDateFormat(dateFormat);

Calendar cal = Calendar.getInstance(); // calendar function to get date and time

int tt = Integer.parseInt(sdf2.format(cal.getTime())); // converting string to integer

if(tt>=8&&tt<=12){

temp=0;

}

else if(tt>12&&tt<=15){

temp=1;

}

else if(tt>15&&tt<=18){

temp=2;

}

else if(tt>18&&tt<=19){

temp=3;

}

else if(tt>19&&tt<=24){

temp=4;

}

else if(tt>=0 &&tt<=5){

temp=4;

}

else if(tt>5&&tt<8){

temp=5;

}

else{

temp=-1;

}

Random rn = new Random();

// variable r gets the random no. from the array between the mentioned range

int r =rn.nextInt(range[(temp\*2)+1]-range[temp\*2]);

r= r + range[temp\*2];

return r;

}

// main function used in order to call the control function

public static void main(String[] args){

demo d1=new demo();

d1.control\_algo();

}

// control function which controls the entire working of system

public void control\_algo(){

// used to store the value of learned intensity from the table

float learned\_intensity = 0;

// used in order to store the current day

String dateFormat=null;

String weekday = null;

int tt = 0;

//used to store the sql query in order insert values into database

String query = null;

/\* infinite loop to measure the intensity of sunlight and

check the count of people in the room.\*/

while(true){

intensity\_input = random\_intensity();

if(count == 0){

//System.out.println("all lights switched off count =0");

old\_intensity = intensity\_input;

//System.out.println(learned\_intensity-old\_intensity);

wait\_algo();

}

// code used to retrieve the current system day

dateFormat="EEEE";

Calendar cal = Calendar.getInstance();

SimpleDateFormat sdf1 = new SimpleDateFormat(dateFormat);

weekday = sdf1.format(cal.getTime());

weekday = weekday.toLowerCase();

// code used to retrieve the current system hour

dateFormat = "H";

SimpleDateFormat sdf2 = new SimpleDateFormat(dateFormat);

tt = Integer.parseInt(sdf2.format(cal.getTime()));

/\* query in order to fetch the learned value from the learned table based on current day\*/

try{

rs = st.executeQuery(query);

while (rs.next())

{

learned\_intensity = Integer.parseInt(rs.getString(1));

//System.out.println(rs.get\_inputString(1));

}

old\_intensity = intensity\_input;

// difference calculated to get the current intensity that is required inside the room

float id = learned\_intensity - intensity\_input;

if(id<0){

id=0;

}

// used to display the entire result

System.out.println( " No. of people in room: " + count + "Input intensity " + intensity\_input + "learned intensity " + learned\_intensity + " Output Set " + id);

wait\_algo();

}

catch (SQLException e1){

System.err.println(e1);

}

}

}

// function which manages the call to other function based on system time

public void wait\_algo(){

while(true){

// code to fetch current system minute in order to call sum\_algo function

String dateFormat1= "mm";

Calendar cal = Calendar.getInstance();

SimpleDateFormat sdf = new SimpleDateFormat(dateFormat1);

int min = Integer.parseInt(sdf.format(cal.getTime()));

// code to fetch current system seconds in order to call avg\_algo function

String dateFormat2= "ss";

SimpleDateFormat sdf1 = new SimpleDateFormat(dateFormat2);

int sec = Integer.parseInt(sdf1.format(cal.getTime()));

// code to fetch current system minute in order to call learn\_algo function

String dateFormat3= "H";

SimpleDateFormat sdf2 = new SimpleDateFormat(dateFormat3);

int hr = Integer.parseInt(sdf2.format(cal.getTime()));

if(hr==0 && min==0 && sec>=0 && sec<4){

learn\_algo();

}

if(sec>=0&&sec<4){

sum\_algo();

}

if(min==0 && sec>=0 && sec<4){

avg\_algo();

}

intensity\_input=random\_intensity();

random\_count();

// checking if the change in intensity is more than 10%

float a1=(float)(old\_intensity \* (102.0/100.0));

float a2=(float)(old\_intensity \* (98.0/100.0));

if (count!=0){

if ( (intensity\_input > a1) || (intensity\_input < a2 )){

try

{

Thread.sleep(4000);

}

catch (InterruptedException ie)

{

//continue;

}

return;

}

}

// checking if there is anyone present inside room

if (count == 0){

System.out.println("all lghts switched off count =0");

try

{

Thread.sleep(4000);

}

catch (InterruptedException ie)

{

}

}

}

}

// function to implement the machine learning

public static void learn\_algo(){

String dateFormat = "K";

String dateFormat1= "EEEE";

// try block to open a connection to database

try{

Class.forName("com.mysql.jdbc.Driver");

db = DriverManager.getConnection(url,usr,pass);

st = db.createStatement();

}

catch (SQLException e11){

System.err.println(e11);

}

catch (ClassNotFoundException e22){

System.err.println(e22);

}

Calendar cal = Calendar.getInstance();

SimpleDateFormat sdf = new SimpleDateFormat(dateFormat1);

SimpleDateFormat sdf1 = new SimpleDateFormat(dateFormat);

String day = sdf.format(cal.getTime());

//used to intially store zero's as the sum of intensity

int sum[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

int hour = Integer.parseInt(sdf1.format(cal.getTime()));

System.out.println(hour);

/\* used to get the feedback value of last 15 days where day is same as current day

and current hour is same as current hour from the feedback table\*/

try{

for (int i=0 ; i<24 ; i++){ // for all 24 hour

for(int k=1;k<=15;k++){ // for last 15 days where day is equal to current day

int x=0;

//query used to fetch data from table

// to execute the query

rs = st.executeQuery(query);

//used to calculate the sum

while (rs.next())

{

x=rs.getInt(1);

sum[i] += x;

}

}

}

day = day.toLowerCase();

System.out.println(day);

// loop used to store the the value in learned table based on current hour

for (int i = 0; i< 24; i++){

System.out.println(sum[i]);

// query to update learned table

query = "update ml\_table set lv" + (hour + 1 + i) + " = " + (sum[i]/15) + "where weekday = " + " \"" + day + "\" " + ";";

int n = st.executeUpdate(query);

}

query = "insert into light\_table (dt,weekday,fd1,fd2,fd3,fd4,fd5,fd6,fd7,fd8,fd9,fd10,fd11,fd12,fd13,fd14,fd15,fd16,fd17,fd18,fd19,fd20,fd21,fd22,fd23,fd24) values(CURRENT\_DATE()," + " \" " + day + " \" " + ",1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1);";

int n = st.executeUpdate(query);

}

catch (SQLException e1){

System.err.println(e1);

}

catch (Exception e2){

System.err.println(e2);

}

}

// function to calculate the sum\_ feedback

public static void sum\_algo(){

Random rn = new Random();

int r = rn.nextInt(6);

knob\_feedback = r;

sum\_feedback = sum\_feedback + intensity\_input + knob\_feedback;

//System.out.println("Sum :" + sum\_feedback);

}

// function to calculate the average light intensity for each hour

public static void avg\_algo(){

// used to fetch current hour

String dateFormat = "H";

Calendar cal = Calendar.getInstance();

SimpleDateFormat sdf = new SimpleDateFormat(dateFormat);

int hour = Integer.parseInt(sdf.format(cal.getTime()));

try{

query = "update light\_table set fd" + (hour+1) + " = " + (sum\_feedback/60) + " where dt = CURRENT\_DATE();";

int n = st.executeUpdate(query);

}

catch (SQLException e1){

System.err.println(e1);

}

catch (Exception e2){

System.err.println(e2);

}

System.out.println("Avg algo :" + sum\_feedback /60);

}

}

**4.5 Results**

**Simulation 1**

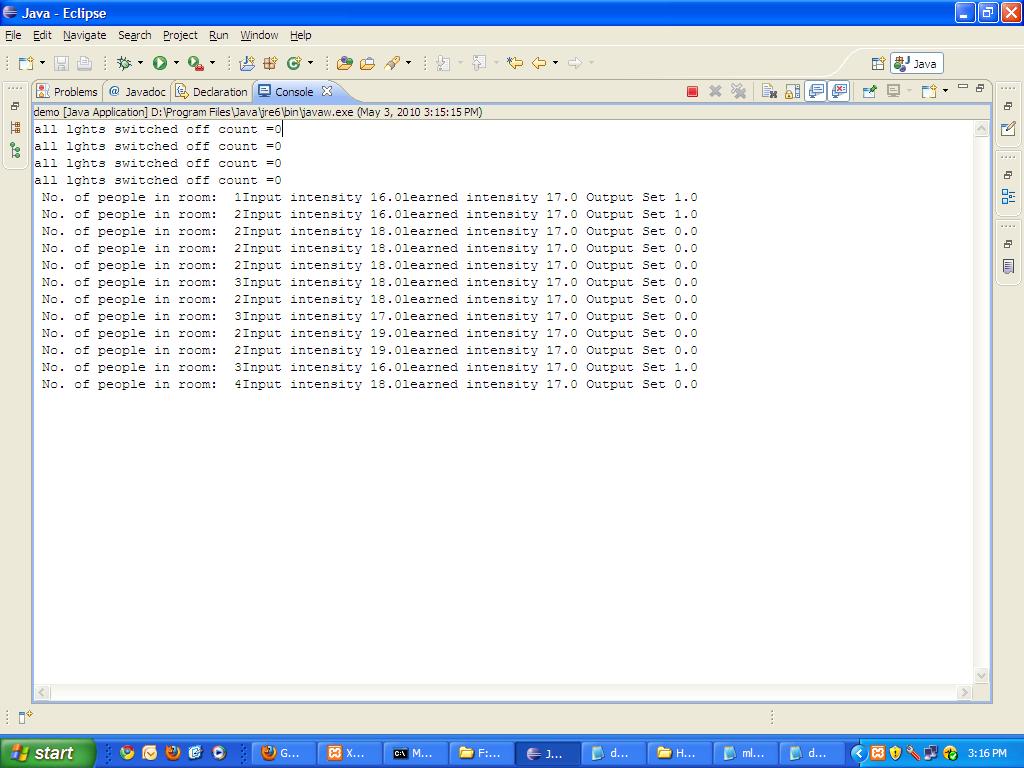
****

Fig 4.1 Simulation 1

This was the result produced during the second run of the program.

**Simulation 2**

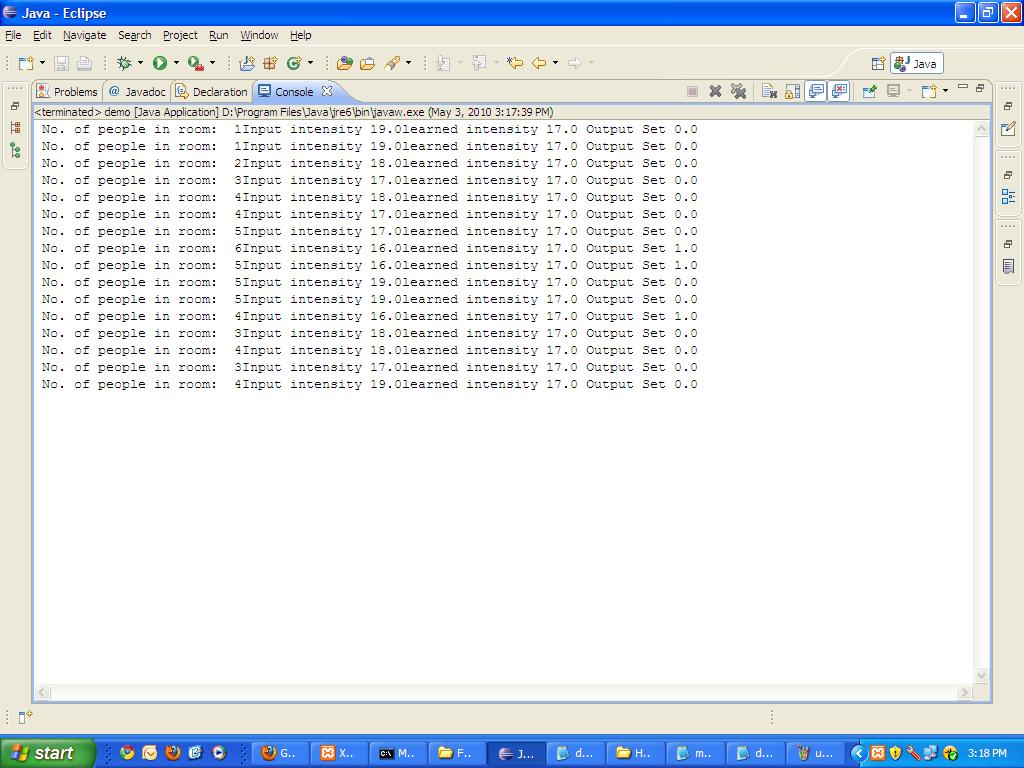
****

Fig 4.2 Simulation 2

**CHAPTER 5**

**CONCLUSION AND SCOPE FOR FUTURE WORK**

**5.1 Conclusion**

The main objective of the system is to conserve energy. The artifact has been built with the motive of conserving energy and making life easier for the user. The working model that we have constructed illustrates this fact.

The software that has been designed by us works for all values possible which allows the user to operate it any desired level of output. But the hardware unit at our disposal has a minor limitation. The output can be varied only in three levels: LOW, MED and HIGH. This is due to that fact that the parallel port we used didn’t have enough pins to show more levels of output. This would have given us more varied levels of brightness in the bulb.

But as shown by the software, it is definitely possible to have varied levels of brightness in the bulb, provided we have enough number of output pins.

Hence we can say that the automated model designed by us is completely user-friendly and saves a lot of energy, money of the user. It can be installed and operated to suit the varied needs of different people, as the system is adaptive. Comforts apart the fact that the system is environment friendly is an added bonus and that alone will convince people to go ahead and implement his system.

**5.2 Innovativeness and Usefulness**

The most unique feature of this system is the implementation of machine learning mechanism. The fact that the system can deduce the pattern of usage and function accordingly makes it an exciting commodity.

Some features and benefits offered by the system are:

* Economical
* Eco friendly
* Less Power Consumption
* Easy to operate and maintain
* Can be used in wide areas

**5.3 Future Scope**

As we have already discussed, the limitation of the system at hand is that it has only three levels of output. This can be overcome by using a decoder if we are not able to provide more than three pins for output. This will enable us to have more levels of output.

A significant change can be brought about to our system by replacing the computer with another component. Since it is only a prototype we have used a computer to maintain the database. But the computer needs to be switched on all the time. This in itself would consume a considerable amount of electricity. This can act as a hindrance to our objective. We can further enhance the efficiency of the system by replacing the computer with a device that does not consume electricity all the time, like a microcontroller. The whole program can be burnt onto the microcontroller and control the whole system by itself.

This prototype can be modified and used according to the needs of the user. We have already seen in section 2.2, the various environments that can make use of this system. In addition to that we can say that a fully automated home can also use this prototype.

Our prototype deals only with the lighting aspects. The same prototype can further developed to include other factors involved in automating a house. We can use a water level indicator to check the amount of water left in the tank. In a similar way we can automate other aspects of a home by automating TV, washing machine and other appliances.

**CHAPTER 6**

**BIBLIOGRAPHY**

1. Artificial Intelligence: A Modern Approach– Stuart Russel, Peter Norvig, 2nd Edition, Pearson Education, 2003.
2. Java - The Complete Reference– Herbert Schildt, 7th Edition, Tata McGraw Hill, 2007
3. http://www.lightingassociates.org/i/u/2127806/f/tech\_sheets/Illuminating\_the\_Classroom\_Enviroment.pdf
4. http://engineering.suite101.com/article.cfm/an\_introduction\_to\_sensors
5. http://en.wikipedia.org/wiki/Sensor