CHAPTER 5
HARDWARES AND SOFTWARES USED

In this project the MSP430 is used as a main controller for the entire control of the system. It is connected with other sensors devices and motors as mentioned in the above lock diagram. The Ultrasonic sensor is used to identify the presence of wastes and also the other Ultrasonic sensor is used to find whether the bin is full or not. If the bin is full then the controller tells the GSM module to send an SMS message to an existing contact with the predefined context.

Then the LCD will show the text that “BIN IS FULL” to prevent the overflow. All the moving parts are controlled by the stepper motor controlled by the L298N motor diver module. The moisture sensor and the inductive proximity sensor is used to identify the metal wastes and food wastes respectively. The entire system is powered by a 12V and 2 ampere power adapter. The basic building blocks of this proposed system are,

- LCD display
- Ultrasonic sensor
- MSP430G2 microcontroller
- NODE MCU module
- DC gear motor
- L298N motor driver
- Inductive proximity sensor
- Moisture sensor

5.1 LCD DISPLAY

A liquid crystal display is a flat panel display, electronic visual display, or video display that uses the light modulating properties of liquid crystals. Liquid crystals do not emit light directly. LCDs are available to display arbitrary images or fixed images which can be displayed or hidden, such as preset words, digits, and 7-segment displays as in a digital clock.

Figure 5.1 LCD display

A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on. A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.

The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. An LCD is made with either a passive matrix or an active matrix display grid. The active matrix LCD is also known as a thin film transistor (TFT) display. The passive matrix LCD has a grid of

conductors with pixels located at each intersection in the grid. A current is sent across two conductors on the grid to control the light for any pixel. An active matrix has a transistor located at each pixel intersection, requiring less current to control the luminance of a pixel. For this reason, the current in an active matrix display can be switched on and off more frequently, improving the screen refresh time.

1. VSS, VDD and VEE

Pin 1 (VSS) is a ground pin and it is certainly needed that this pin should be grounded for LCD to work properly. VEE and VDD are given +5 volts normally. However VEE may have a potentiometer voltage divider network to get the contrast adjusted. But VDD is always at +5V.

2. RS, R/W and E

These three pins are numbered 4, 5 and 6 as shown above. RS is used to make the selection between data and command register.

For RS=0, command register is selected and for RS=1 data register is selected. R/W gives you the choice between writing and reading. If set (R/W=1) reading is enabled. R/W=0 when writing.

Enable pins is used by the LCD to latch information presented to its data pins. When data is supplied to data pins, a high to low pulse must be applied to this pin in-order for the LCD to latch in the data present at the data pins. It may be noted here that the pulse must be of minimum 450ns wide.

3. D0-D7

The 8-bit data pins, D0-D7, are used to send information to the LCD or read the contents of LCD's internal register.

5.2 ULTRASONIC SENSOR

Ultrasonic sensors (also known as transceivers when they both send and receive, but more generally called transducers) work on a principle similar to

radar or sonar which evaluate attributes of a target by interpreting the echoes from radio or sound waves respectively. Ultrasonic sensors generate high frequency sound waves and evaluate the echo which is received back by the sensor.

Sensors calculate the time interval between sending the signal and receiving the echo to determine the distance to an object. This technology can be used for measuring wind speed and direction (anemometer), tank or channel level, and speed through air or water. For measuring speed or direction a device uses multiple detectors and calculates the speed from the relative distances to particulates in the air or water. To measure tank or channel level, the sensor measures the distance to the surface of the fluid. Further applications include: humidifiers, sonar, medical ultrasonography, burglar alarms and non-destructive testing.

Systems typically use a transducer which generates sound waves in the ultrasonic range, above 18,000 hertz, by turning electrical energy into sound, then upon receiving the echo turn the sound waves into electrical energy which can be measured and displayed. The technology is limited by the shapes of surfaces and the density or consistency of the material. Foam, in particular, can distort surface level readings.

Ultrasonic transducers or ultrasonic sensors are a type of acoustic sensor divided into three broad categories: transmitters, receivers and transceivers. Transmitters convert electrical signals into ultrasound, receivers convert ultrasound into electrical signals, and transceivers can both transmit and receive ultrasound.

In a similar way to radar and sonar, ultrasonic transducers are used in systems which evaluate targets by interpreting the reflected signals. For example, by measuring the time between sending a signal and receiving an

echo the distance of an object can be calculated. Passive ultrasonic sensors are basically microphones that detect ultrasonic noise that is present under certain conditions.

Ultrasound can be used for measuring wind speed and direction (anemometer), tank or channel fluid level, and speed through air or water. For measuring speed or direction, a device uses multiple detectors and calculates the speed from the relative distances to particulates in the air or water. To measure tank or channel liquid level, and also sea level (tide gauge), the sensor measures the distance (ranging) to the surface of the fluid. Further applications include: humidifiers, sonar, medical ultrasonography, burglar alarms, non-destructive testing and wireless charging.

Figure 5.2 Ultrasonic sensor

Systems typically use a transducer which generates sound waves in the ultrasonic range, above 18 kHz, by turning electrical energy into sound, then upon receiving the echo turn the sound waves into electrical energy which can be measured and displayed. The technology is limited by the shapes of surfaces and the density or consistency of the material. Foam, in particular, can distort surface level readings. This technology, as well, can detect approaching objects and track their positions.

Ultrasound can also be used to make point-to-point distance measurements by transmitting and receiving discrete bursts of ultrasound between transducers. This technique is known as Sonomicrometry where the transit-time of the ultrasound signal is measured electronically (i.e. digitally)

and converted mathematically to the distance between transducers assuming the speed of sound of the medium between the transducers is known.

This method can be very precise in terms of temporal and spatial resolution because the time-of-flight measurement can be derived from tracking the same incident (received) waveform either by reference level or zero crossing. This enables the measurement resolution to far exceed the wavelength of the sound frequency generated by the transducers.

5.3 MSP430G2 MICROCONTROLLER

The Texas Instruments MSP430 family of ultra-low-power microcontrollers consists of several devices featuring different sets of peripherals targeted for various applications. The architecture, combined with five low-power modes, is optimized to achieve extended battery life in portable measurement applications. The device features a powerful 16-bit RISC CPU, 16-bit registers, and constant generators that contribute to maximum code efficiency. The digitally controlled oscillator (DCO) allows wake-up from low-power modes to active mode in less than 1 µs.

The MSP430G2x13 and MSP430G2x53 series are ultra-low-power mixed signal microcontrollers with built-in 16-bit timers, up to 24 I/O capacitive-touch enabled pins, a versatile analog comparator, and built-in communication capability using the universal serial communication interface. In addition the MSP430G2x53 family members have a 10-bit analog-to-digital (A/D) converter. Typical applications include low-cost sensor systems that capture analog signals, convert them to digital values, and then process the data for display or for transmission to a host system.

A microcontroller (abbreviated MCU or µC) is a computer system on a chip that does a job. It contains an integrated processor, memory(a small amount of RAM, program memory, or both), and programmable input/output peripherals, which are used to interact with things connected to the chip. A microcontroller is different than a microprocessor, which only contains a CPU (the kind used in a Personal Computer).

First released in 1971 by the Intel company, microcontrollers began to become popular in their first few years. The extremely useful Intel 8008 microprocessor was then released, but it was still impractical because of high cost for each chip. These first microcontrollers combined different types of computer memory on one unit. After people began to see how useful they were, micro controllers were constantly being upgraded, with people trying to find new ways to make them better. Cost was reduced over time and by the early 2000s, micro controllers were widely used across the world.

Other terms for a microcontroller are embedded system and embedded controller, because the microcontroller and its support circuits are often built into, or embedded in, a single chip. In addition to the usual arithmetic and logic elements of a general microprocessor, the microcontroller also has additional elements such as RAM for data storage, read-only memory for program storage, flash memory for permanent data storage, and other devices (peripherals).

Microcontrollers often operate at very low speed compared to microprocessors (at clock speeds of as little as 32 kHz), but this is useful for typical applications. They also consume very little power (mill watts or even micro watts). Microcontrollers are used in automatic products and devices, such as car engine systems, remote controls, machines, appliances, power tools, and toys. These are called embedded systems. Microcontrollers

can also be found at work in solar power and energy harvesting, anti-lock braking systems in cars, and have many uses in the medical field as well.

The TI MSP430™ microcontroller (MCU) portfolio offers a wide variety of 16-bit MCUs with ultra-low-power and integrated analog and digital peripherals for sensing and measurement applications. MSP430 MCUs are supported by development kits, reference designs, software, training, documentation and online support to get you from concept to prototype to production quickly.

The MSP430 is a family of 16-bit RISC microcontrollers produced by Texas Instruments. The MSP430 microcontroller was developed at Texas Instruments in 1993. At the beginning Texas Instruments only offered the MSP430 in Europe. Since 1997 the MSP430 microcontroller family is offered worldwide. The most important feature of the MSP430 is its low power consumption. However, the flexibility of its peripheral modules and the easy way to use it is the reason why this microcontroller is also used as a general purpose microcontroller.

There are six general generations of MSP430 processors. In order of development, they were the '3xx generation, the '1xx generation, the '4xx generation, the '2xx generation, the '5xx generation, and the '6xx generation. The digit after the generation identifies the model (generally higher model numbers are larger and more capable), the third digit identifies the amount of memory on board, and the fourth, if present, identifies a minor model variant. The most common variation is a different on-chip analog-to-digital converter. The 3xx and 1xx generations were limited to a 16-bit address space. In the later generations this was expanded to include '430X' instructions that allow a 20-bit address space.

As happened with other processor architectures (e.g. the processor of the PDP-11), extending the addressing range beyond the 16-bit word size introduced some peculiarities and inefficiencies for programs larger than 64 kBytes. In the following list, it helps to think of the typical 200 mA·Hr capacity

of a CR2032 lithium coin cell as 200,000 μA·Hr, or 22.8 μA·year. Thus, considering only the CPU draw, such a battery could supply a 0.7 μA current draw for 32 years. (In reality, battery self-discharge would reduce this number.)

The significance of the 'RAM retention' vs. the 'real-time clock mode' is that in real time clock mode the CPU can go to sleep with a clock running which will wake it up at a specific future time. In RAM retention mode, some external signal is required to wake it, e.g. I/O pin signal or SPI slave receive interrupt. The peripheral is not needed, the pin may be used for general purpose I/O. The pins are divided into 8-bit groups called "ports", each of which is controlled by a number of 8-bit registers. In some cases, the ports are arranged in pairs which can be accessed as 16-bit registers.

The MSP430 family defines 11 I/O ports, P0 through P10, although no chip implements more than 10 of them. P0 is only implemented on the '3xx family. P7 through P10 are only implemented on the largest members (and highest pin count versions) of the '4xx and '2xx families. The newest '5xx and '6xx families has P1 through P11, and the control registers are reassigned to provide more port pairs. Each port is controlled by the following registers. Ports which do not implement particular features (such as interrupt on state change) do not implement the corresponding registers.

1. PxIN

Port x input. This is a read-only register, and reflects the current state of the port's pins.

2. PxOUT

Port x output. The values written to this read/write register are driven out the corresponding pins when they are configured to output.

3. PxDIR

Port x data direction. Bits written as 1 configure the corresponding pin for output. Bits written as 0 configure the pin for input.

4. PxSEL

Port x function select. Bits written as 1 configure the corresponding pin for use by the specialized peripheral. Bits written as 0 configure the pin for general purpose I/O. Port 0 ('3xx parts only) is not multiplexed with other peripherals and does not have a P0SEL register.

5. PxREN

Port x resistor enable ('2xx & '5xx only). Bits set in this register enable weak pull-up or pull-down resistors on the corresponding I/O pins even when they are configured as inputs. The direction of the pull is set by the bit written to the PxOUT register.

6. PxDS

Port x drive strength ('5xx only). Bits set in this register enable high current outputs. This increases output power, but may cause EMI. Ports (0–2) can produce interrupts when inputs change. Additional registers configure this ability.

7. PxIES

Port x interrupt edge select. Selects the edge which will cause the PxIFG bit to be set. When the input bit changes from matching the PxIES state to not matching it (i.e. whenever a bit in PxIES XOR PxIN changes from clear to set), the corresponding PxIFG bit is set.

8. PxIE

Port x interrupt enable. When this bit and the corresponding PxIFG bit are both set, an interrupt is generated.

9. PxIFG

Port x interrupt flag. Set whenever the corresponding pin makes the state change requested by PxIES. Can be cleared only by software. (Can also be set by software.)

10. PxIV

Port x interrupt vector ('5xx only). This 16-bit register is a priority encoder which can be used to handle pin-change interrupts. If n is the lowest-numbered interrupt bit which is pending in PxIFG and enabled in PxIE, this register reads as 2n+2. If there is no such bit, it reads as 0. The scale factor of 2 allows direct use as an offset into a branch table. Reading this register also clears the reported PxIFG flag.

Some pins have special purposes either as inputs or outputs. In this case, the PxDIR bit controls which of the two functions the pin performs when the PxSEL bit is set. If there is only one special function, then PxDIR is generally ignored. The PxIN register is still readable if the PxSEL bit is set, but interrupt generation is disabled. If PxSEL is clear, the special function's input is frozen and disconnected from the external pin. Also, configuring a pin for general purpose output does not disable interrupt generation. Some of the peripherals are as follows.

5.3.1 ANALOG

1. Analog-to-digital converter

The MSP430 line offers two types of analog-to-digital conversion (ADC). 10- and 12-bit successive approximation converters, as well as a 16-bit Sigma-Delta converter. Data transfer controllers and a 16 word conversion-and-control buffer allow the MSP430 to convert and store samples without CPU intervention, minimizing power consumption.

2. Analog pool

The Analog Pool (A-POOL) module can be configured as an ADC, DAC, comparator, SVS or temperature sensor. It allows flexibility for the user to program a series of analog functions with only one setup.

3. Comparator A, A+

The MSP430's comparator module provides precision slope Analog-to-Digital Conversions. Monitors external analog signals and provides voltage and resistor value measurement. Capable of selectable power modes.

4. DAC12

The DAC12 module is a 12-bit, voltage-output DAC featuring internal/external reference selection and programmable settling time for optimal power consumption. It can be configured in 8- or 12-bit mode. When multiple DAC12 modules are present, they may be grouped together for synchronous update operation.

5. Op Amps

Feature single supply, low current operation with rail-to-rail outputs and programmable settling times. Software selectable configuration options: unity gain mode, comparator mode, inverting PGA, non-inverting PGA, differential and instrumentation amplifier.

6. Sigma Delta (SD)

The SD16/SD16_A/SD24_A modules each feature 16-/24-bit sigma-delta A/D converters with an internal 1.2-V reference. Each converter has up to eight fully differential multiplexed inputs, including a built-in temperature sensor. The converters are second-order oversampling sigma-delta modulators with selectable oversampling ratios of up to 1024 (SD16_A/SD24_A) or 256 (SD16).

5.3.2 TIMERS

1. Basic timer (BT)

The BT has two independent 8-bit timers that can be cascaded to form a 16-bit timer/counter. Both timers can be read and written by software. The BT is extended to provide an integrated RTC. An internal calendar compensates for months with less than 31 days and includes leap-year correction.

2. Real-Time Clock

RTC_A/B are 32-bit hardware counter modules that provide clock counters with a calendar, a flexible programmable alarm, and calibration. The RTC_B includes a switchable battery backup system that provides the ability for the RTC to operate when the primary supply fails.

3. 16-bit timers

Timer_A, Timer_B and Timer_D are asynchronous 16-bit timers/counters with up to seven capture/compare registers and various operating modes. The timers support multiple capture/compares, PWM outputs, and interval timing. They also have extensive interrupt capabilities. Timer_B introduces additional features such as programmable timer lengths (8, 10, 12 or 16-bit) and double-buffered compare register updates, while Timer_D introduces a high-resolution mode (4 ns resolution).

4. Watchdog (WDT+)

The WDT+ performs a controlled system restart after a software problem occurs. If the selected time interval expires, a system reset is generated. If the watchdog function is not needed in an application, the module can be configured as an interval timer and can generate interrupts at selected time intervals.

5.3.3 SYSTEM

1. Advanced Encryption Standard (AES)

The AES accelerator module performs encryption and decryption of 128-bit data with 128-bit keys according to the advanced encryption standard in hardware, and can be configured with user software.

2. Brown-Out Reset (BOR)

The BOR circuit detects low supply voltages and resets the device by triggering a power-on reset (POR) signal when power is applied or removed. The MSP430 MCU’s zero-power BOR circuit is continuously turned on, including in all low-power modes.

3. Direct Memory Access (DMA) Controller

The DMA controller transfers data from one address to another across the entire address range without CPU intervention. The DMA increases the throughput of peripheral modules and reduces system power consumption. The module features up to three independent transfer channels.

Although the MSP430's DMA subsystem is very capable it has several flaws, the most significant of which is the lack of an external transfer strobe. Although a DMA transfer can be triggered externally, there is no external indication of completion of a transfer. Consequently DMA to and from external sources is limited to external trigger per byte transfers, rather than full blocks automatically via DMA. This can lead to significant complexity (as in requiring extensive hand tweaking of code) when implementing processor to processor or processor to USB communications.

The reference cited uses an obscure timer mode to generate high speed strobes for DMA transfers. Unfortunately, the timers are not flexible enough to easily make up for the lack of an external DMA transfer strobe.

DMA operations that involve word transfers to byte locations cause truncation to 8 bits rather than conversion to two byte transfers. This makes DMA with A/D or D/A 16 bit values less useful than it could be (although it is possible to DMA these values through port A or B on some versions of the MSP 430 using an externally visible trigger per transfer such as a timer output).

4. Enhanced Emulation Module (EEM)

The EEM provides different levels of debug features such as 2-8 hardware breakpoints, complex breakpoints, break when read/write occurs at specified address, and more. Embedded into all flash-based MSP430 devices.

5. Hardware multiplier

Some MSP430 models include a memory-mapped hardware multiplier peripheral which performs various 16×16+32→33-bit multiply-accumulate operations. Unusually for the MSP430, this peripheral does include an implicit 2-bit write-only register, which makes it effectively impossible to context switch. This peripheral does not interfere with CPU activities and can be accessed by the DMA.

The address written determines the operation performed. While the value written can be read back from any of the registers, the register number written to cannot be recovered. Then, each time a write is performed to the OP2 register, a multiply is performed and the result stored or added to the result registers. The Sum Ext register is a read-only register that contains the carry out of the addition (0 or 1) in case of an unsigned multiply), or the sign extension of the 32-bit sum (0 or -1) in case of a signed multiply. In the case of a signed multiply-accumulate, the Sum Ext value must be combined with the most significant bit of the prior SumHi contents to determine the true carry out result (-1, 0, or +1).

The result is available after three clock cycles of delay, which is the time required to fetch a following instruction and a following index word. Thus, the

delay is typically invisible. An explicit delay is only required if using an indirect addressing mode to fetch the result.

6. Memory Protection Unit (MPU)

The FRAM MPU protects against accidental writes to designated read-only memory segments or execution of code from a constant memory. The MPU can set any portioning of memory with bit level addressing, making the complete memory accessible for read, write and execute operations in FRAM devices.

7. Power management module (PMM)

The PMM generates a supply voltage for the core logic, and provides several mechanisms for the supervision and monitoring of both the voltage applied to the device and the voltage generated for the core. It is integrated with a low-dropout voltage regulator (LDO), brown-out reset (BOR), and a supply voltage supervisor and monitor.

8. Supply-Voltage Supervisor (SVS)

The SVS is a configurable module used to monitor the AVCC supply voltage or an external voltage. The SVS can be configured to set a flag or generate a power-on reset (POR) when the supply voltage or external voltage drops below a user-selected threshold.

5.3.4 COMMUNICATION AND INTERFACE

1. Capacitive Touch Sense I/Os

The integrated capacitive touch sense I/O module offers several benefits to touch button and touch slider applications. The system does not require external components to create the self-oscillation (reducing bill of materials) and the capacitor (that defines the frequency of the self-oscillation) can be connected directly. In addition, there is no need for external MUXes to allow multiple pads and each I/O pad can directly serve as a cap sense input. A

hysteresis of ~0.7 V ensures robust operation. Control and sequencing is done completely in software.

2. General Purpose I/Os

MSP430 devices have up to 12 digital I/O ports implemented. Each port has eight I/O pins. Every I/O pin can be configured as either input or output, and can be individually read or written to. Ports P1 and P2 have interrupt capability. MSP430F2xx, F5xx and some F4xx devices feature built-in, individually configurable pull-up or pull-down resistors.

3. Sub-GHz RF Front End

The flexible CC1101 sub-1 GHz transceiver delivers the sensitivity and blocking performance required to achieve successful communication links in any RF environment. It also features low current consumption and supports flexible data rates and modulation formats.

4. USART (UART, SPI, I²C)

The universal synchronous/asynchronous receive/transmit (USART) peripheral interface supports asynchronous RS-232 and synchronous SPI communication with one hardware module. The MSP430F15x/16x USART modules also support I²C, programmable baud rate, and independent interrupt capability for receive and transmit.

5. USB

The USB module is fully compliant with the USB 2.0 specification and supports control, interrupt and bulk transfers at a data rate of 12 Mbps (full speed). The module supports USB suspend, resume and remote wake-up operations and can be configured for up to eight input and eight output endpoints. The module includes an integrated physical interface (PHY); a phase-locked loop (PLL) for USB clock generation; and a flexible power-supply system enabling bus-powered and self-powered devices.

6. USCI (UART, SPI, I²C, LIN, IrDA)

The universal serial communication interface (USCI) module features two independent channels that can be used simultaneously. The asynchronous channel (USCI_A) supports UART mode; SPI mode; pulse shaping for IrDA; and automatic baud-rate detection for LIN communications. The synchronous channel (USCI_B) supports I²C and SPI modes.

7. USI (SPI, I²C)

The universal serial interface (USI) module is a synchronous serial communication interface with a data length of up to 16-bits and can support SPI and I²C communication with minimal software.

8. Infrared Modulation

Available on the MSP430FR4xxx and MSP430FR2xxx series chips, this feature is configured via the SYSCFG register set. This peripheral ties into other peripherals (Timers, eUSCI_A) to generate an IR modulated signal on an output pin.

5.3.5 METERING

1. ESP430 (integrated in FE42xx devices)

The ESP430CE module performs metering calculations independent of the CPU. Module has separate SD16, HW multiplier, and the ESP430 embedded processor engine for single-phase energy-metering applications.

2. Scan Interface (SIF)

The SIF module, a programmable state machine with an analog front end, is used to automatically measure linear or rotational motion with the lowest possible power consumption. The module features support for different types of LC and resistive sensors and for quadrature encoding.

5.3.6 DISPLAY

1. LCD/LCD_A/LCD_B

The LCD/LCD_A controller directly drives LCDs for up to 196 segments. Supports static, 2-mux, 3-mux, and 4-mux LCDs. LCD_A module has integrated charge pump for contrast control. LCD_B enables blinking of individual segments with separate blinking memory.

2. LCD_E

The LCD_E controller comes with the newer MSP430FR4xxx series microcontrollers and directly drives LCDs up to 448 segments. Supports static, 2-mux, 3-mux, 4-mux, 5-mux, 6-mux, 7-mux, 8-mux (1/3 bias) LCDs. Segment and Common pins may be reprogrammed to available LCD drive pins. This peripheral may be driven in LPM3.5 (RTC running+Main CPU core shutdown low-power mode).

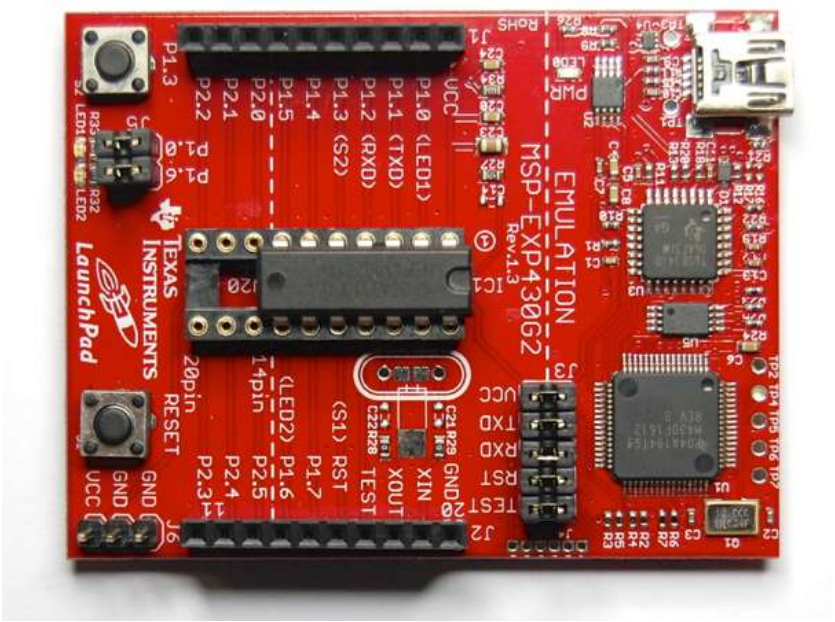

Figure 5.3 MSP430G2 Microcontroller

5.3.7 MSP430G2 ARCHITECTURE

The microcontroller's performance is directly related to the 16-bit data bus, the 7 addressing modes and the reduced instructions set, which allows a

shorter, denser programming code for fast execution. These microcontroller families share a 16-bit CPU (Central Processing Unit) core, RISC1 type, intelligent peripherals, and flexible clock system that interconnects using a Von Neumann2 common memory address bus (MAB) and memory data bus (MDB) architecture.

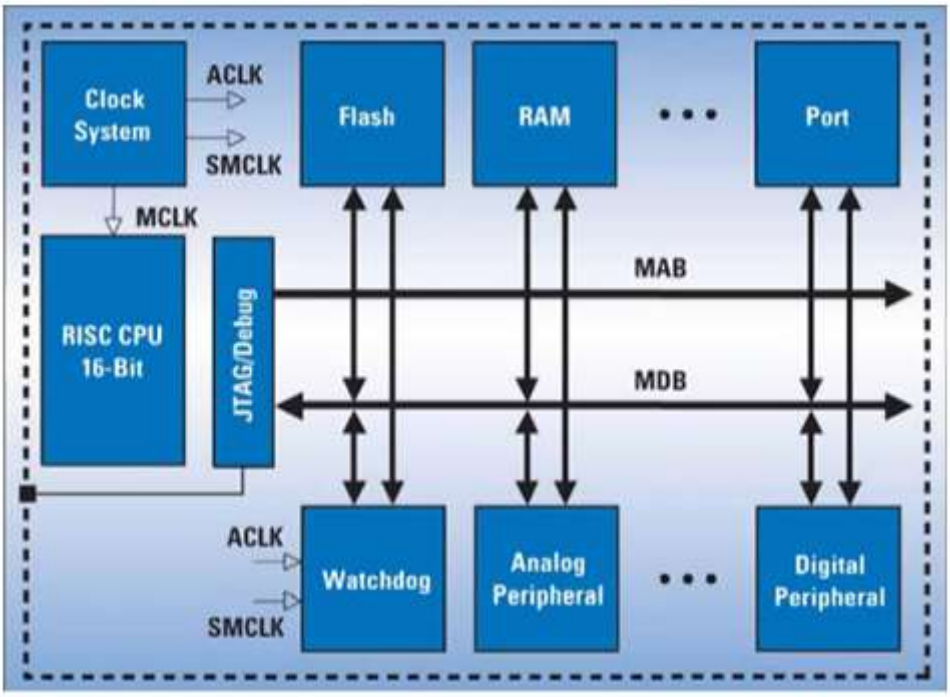

Figure 5.4 MSP430G2 Architecture

5.3.8 WORKING OF MSP430G2

The MSP430 can be used for low powered embedded devices. The current drawn in idle mode can be less than 1 µA. The top CPU speed is 25 MHz. It can be throttled back for lower power consumption. The MSP430 also uses six different low-power modes, which can disable unneeded clocks and CPU. Additionally, the MSP430 is capable of wake-up times below 1 microsecond, allowing the microcontroller to stay in sleep mode longer, minimizing its average current consumption. The device comes in a variety of configurations featuring the usual peripherals such as follows.

Internal oscillator, timer including PWM, watchdog, USART, SPI, I²C, 10/12/14/16/24-bit ADCs, and brownout reset circuitry. Some less usual peripheral options include comparators (that can be used with the timers to do simple ADC), on-chip op-amps for signal conditioning, 12-bit DAC, LCD driver, hardware multiplier, USB, and DMA for ADC results. Apart from some older EPROM(MSP430E3xx) and high volume mask ROM (MSP430Cxxx) versions, all of the devices are in-system programmable via JTAG (full four-wire or Spy-Bi-Wire) or a built in bootstrap loader (BSL) using UART such as RS232, or USB on devices with USB support.

There are, however, limitations that preclude its use in more complex embedded systems. The MSP430 does not have an external memory bus, so it is limited to on-chip memory (up to 512 KB flash memory and 66 KB RAM) which may be too small for applications that require large buffers or data tables. Also, although it has a DMA controller, it is very difficult to use it to move data off the chip due to a lack of a DMA output strobe.

5.3.9 PROGRAMMING IN MSP430G2

Figure 5.5 Programming in MSP430G2

Before we start anything TI would have already uploaded a sample Program on your MSP430G2553 Microcontroller, so let us power the board and check if it is working. You can power board through the mini USB jack and once you do it, you should notice the LEDs (red and green) at the bottom

left corner of your board glowing alternatively. You can then press the push button connected to P1.3 to check if the internal temperature sensor is working.

```
#define LED_PIN 13                // Pin number attached to LED.

Void setup () {
   Pin Mode (LED_PIN, OUTPUT);      // Configure pin 13 to be a digital output.
}

void loop() {
   digitalWrite(LED_PIN, HIGH);    // Turn on the LED.
   delay(1000);                    // Wait 1 second (1000 milliseconds).
   digitalWrite(LED_PIN, LOW);     // Turn off the LED.
   delay(1000);                    // Wait 1 second.
}
```

This program uses the functions pinMode(), digitalWrite(), and delay(), which are provided by the internal libraries included in the IDE environment. The program is usually loaded in the Arduino by the manufacture.

5.4 DC GEAR MOTOR

A DC motor is any of a class of rotary electrical machines that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current flow in part of the motor.

DC motors were the first type widely used, since they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances.

Fig 5.6 Dc Gear Motor

The universal motor can operate on direct current but is a lightweight brushed motor used for portable power tools and appliances. Larger DC motors are used in propulsion of electric vehicles, elevator and hoists, or in drives for steel rolling mills. The advent of power electronics has made replacement of DC motors with AC motors possible in many applications.

5.5 L298N MOTOR DRIVER

The L298N is an integrated monolithic circuit in a 15- lead Multi-watt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver de-signed to accept standard TTL logic level sand drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the in-put signals .The emitters of the lower transistors of each bridge are connected together rand the corresponding external terminal can be used for the connection of an

external sensing resistor. An additional Supply input is provided so that the logic works at a lower voltage.

Motor drives are circuits used to run a motor. In other words, they are commonly used for motor interfacing. These drive circuits can be easily interfaced with the motor and their selection depends upon the type of motor being used and their ratings (current, voltage). Motor Driver circuits are current amplifiers.

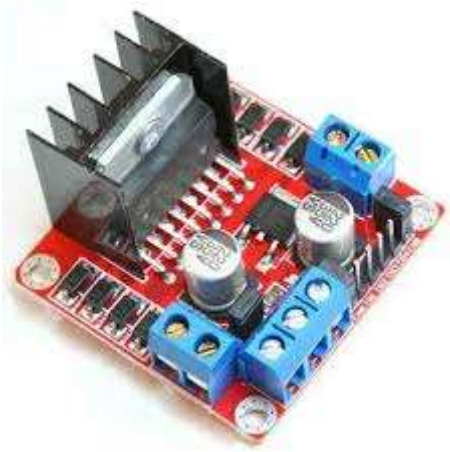

Fig 5.7 L298N driver

They act as a bridge between the controller and the motor in a motor drive. Motor drivers are made from discrete components which are integrated inside an IC. The input to the motor driver IC or motor driver circuit is a low current signal. The function of the circuit is to convert the low current signal to a high current signal.

This high current signal is then given to the motor. The motor can be a brushless DC motor, brushed DC motor, stepper motor, other DC motors etc. In other words we can say the output of the controller or processor is not enough to drive a motor. In such a case direct interfacing of controllers to the motor is not possible. A small motor can be started by simply plugging it into an electrical receptacle or by using a switch or circuit breaker. A larger motor requires a specialized switching unit called a motor starter or motor contactor. When energized, a direct on line (DOL) starter immediately connects the motor

terminals directly to the power supply. Reduced-voltage, star-delta or soft starters connect the motor to the power supply through a voltage reduction device and increases the applied voltage gradually or in steps.

In smaller sizes a motor starter is a manually operated switch; larger motors, or those requiring remote or automatic control, use magnetic contactors. Very large motors running on medium voltage power supplies (thousands of volts) may use power circuit breakers as switching elements. A direct on line (DOL) or across the line starter applies the full line voltage to the motor terminals, the starters or cubicle locations, can usually be found on an ELO drawing. This is the simplest type of motor starter. A DOL motor starter also contains protection devices, and in some cases, condition monitoring.

Smaller sizes of direct on-line starters are manually operated; larger sizes use an electromechanical contactor (relay) to switch the motor circuit. Solid-state direct on line starters also exist. A direct on line starter can be used if the high inrush current of the motor does not cause excessive voltage drop in the supply circuit. The maximum size of a motor allowed on a direct on line starter may be limited by the supply utility for this reason. For example, a utility may require rural customers to use reduced-voltage starters for motors larger than 10 kW. DOL starting is sometimes used to start small water pumps, compressors, fans and conveyor belts.

In the case of an asynchronous motor, such as the 3-phase squirrel-cage motor, the motor will draw a high starting current until it has run up to full speed. This starting current is typically 6-7 times greater than the full load current. To reduce the inrush current, larger motors will have reduced-voltage starters or variable speed drives in order to minimize voltage dips to the power supply. A reversing starter can connect the motor for rotation in either direction. Such a starter contains two DOL circuits—one for clockwise operation and the other for

counter-clockwise operation, with mechanical and electrical interlocks to prevent simultaneous closure. For three phase motors, this is achieved by swapping the wires connecting any two phases.

5.5.1 OPERATION

One of the annoying features of the unit is the lack of internal parasitic (flywheel) diodes to deal with voltage spikes. D1 - D8 are used for this purpose. They can be 1N5819 Schottky diodes. I have used more common 1N4001 rectifier diodes and they seem to work fine. Perhaps an updated version will include these internally.

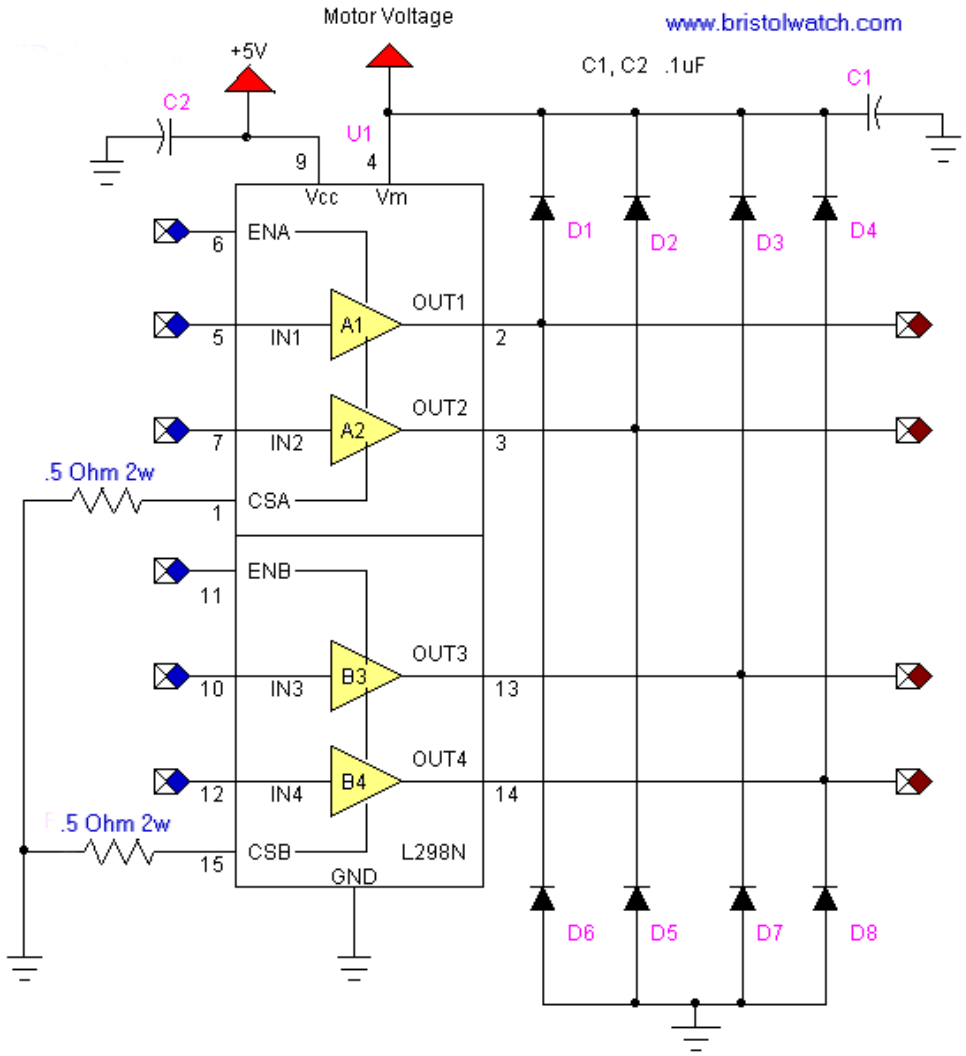

Fig 5.8` L298N Driver Circuit

The four power amplifiers and grouped in pairs of two with individual enable pins (ENA, ENB) and individual current sense pins (CSA, CSB) for each pair. The current sense pins in general can be tied to ground, but one can insert low value resistor, whose voltage reading is proportional to current.

ENA, ENB, and In1-In4 are all standard 5-volt TTL logic making connection to most micro-controllers easy. ENA will turn on A1 and A2 when with a digital HIGH (5-volts) and off when LOW (0 volts); the corresponding outputs will be floating when off. Same is true of ENB, In3 and In4. ENA and ENB can be connected directly together to enable both channels at once or simply tied to +5 volts and both channels making all four outputs active at all times. A 5-volt TTL level input to In1, In2 In3, or In4 will produce a corresponding output of Vm (motor voltage) minus about a volt.

5.6 INDUCTIVE PROXIMITY SENSOR

Proximity sensors detect the presence or absence of objects using electromagnetic fields, light, and sound. There are many types, each suited to specific applications and environments.

These non-contact proximity sensors detect ferrous targets, ideally mild steel thicker than one millimeter. They consist of four major components: a ferrite core with coils, an oscillator, a Schmitt trigger, and an output amplifier. The oscillator creates a symmetrical, oscillating magnetic field that radiates from the ferrite core and coil array at the sensing face. When a ferrous target enters this magnetic field, small independent electrical currents called eddy currents are induced on the metal's surface.

This changes the reluctance (natural frequency) of the magnetic circuit, which in turn reduces the oscillation amplitude. As more metal enters the sensing field the oscillation amplitude shrinks, and eventually collapses. (This

is the “Eddy Current Killed Oscillator” or ECKO principle.) The Schmitt trigger responds to these amplitude changes, and adjusts sensor output. When the target finally moves from the sensor’s range, the circuit begins to oscillate again, and the Schmitt trigger returns the sensor to its previous output.

If the sensor has a normally open configuration, its output is an on signal when the target enters the sensing zone. With normally closed, its output is an off signal with the target present. Output is then read by an external control unit (e.g. PLC, motion controller, smart drive) that converts the sensor on and off states into useable information. Inductive sensors are typically rated by frequency, or on/off cycles per second.

Fig 5.9 Inductive Proximity Sensor

Their speeds range from 10 to 20 Hz in ac, or 500 Hz to 5 kHz in dc. Because of magnetic field limitations, inductive sensors have a relatively narrow sensing range — from fractions of millimeters to 60 mm on average — though longer-range specialty products are available. To accommodate close ranges in the tight confines of industrial machinery, geometric and mounting styles available include shielded (flush), unshielded (non-flush), tubular, and rectangular “flat-pack”. Tubular sensors, by far the most popular, are available with diameters from 3 to 40 mm.

5.7 MOISTURE SENSOR

The Moisture sensor is used to measure the water content (moisture) of soil. When the soil is having water shortage, the module output is at high level, else the output is at low level.

Soil moisture sensors measure the volumetric water content in soil. Since the direct gravimetric measurement of free soil moisture requires removing, drying, and weighting of a sample, soil moisture sensors measure the volumetric water content indirectly by using some other property of the soil, such as electrical resistance, dielectric constant, or interaction with neutrons, as a proxy for the moisture content.

The relation between the measured property and soil moisture must be calibrated and may vary depending on environmental factors such as soil type, temperature, or electric conductivity. Reflected microwave radiation is affected by the soil moisture and is used for sensing in hydrology and agriculture. Portable probe instruments can be used by farmers or gardeners.

Soil moisture sensors typically refer to sensors that estimate volumetric water content. Another class of sensors measure another property of moisture in soils called water potential; these sensors are usually referred to as soil water potential sensors and include tensiometers and gypsum blocks. Technologies commonly used to indirectly measure volumetric water content (soil moisture) include)

- Frequency Domain Reflectometry (FDR): The dielectric constant of a certain volume element around the sensor is obtained by measuring the operating frequency of an oscillating circuit.
- Time Domain Transmission (TDT) and Time Domain Reflectometry (TDR): The dielectric constant of a certain volume element

around the sensor is obtained by measuring the speed of propagation along a buried transmission line.

- Neutron moisture gauges: The moderator properties of water for neutrons are utilized to estimate soil moisture content between a source and detector probe.
- Soil resistivity: Measuring how strongly the soil resists the flow of electricity between two electrodes can be used to determine the soil moisture content.
- Galvanic cell: The amount of water present can be determined based on the voltage the soil produces because water acts as an electrolyte and produces electricity. The technology behind this concept is the galvanic cell.

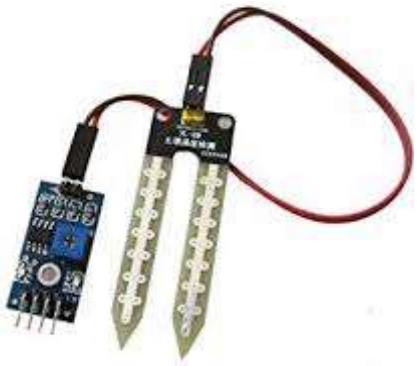

Fig 5.10 Moisture Sensor

This sensor reminds the user to water their plants and also monitors the moisture content of soil. It has been widely used in agriculture, land irrigation and botanical gardening. The Soil Moisture Sensor uses capacitance to measure dielectric permittivity of the surrounding medium. In soil, dielectric permittivity is a function of the water content. The sensor creates a voltage proportional to the dielectric permittivity, and therefore the water content of the soil.

The sensor averages the water content over the entire length of the sensor. There is a 2 cm zone of influence with respect to the flat surface of the sensor,

but it has little or no sensitivity at the extreme edges. The Soil Moisture Sensor is used to measure the loss of moisture over time due to evaporation and plant uptake, evaluate optimum soil moisture contents for various species of plants, monitor soil moisture content to control irrigation in greenhouses and enhance bottle biology experiments.

5.8 NODE MCU

NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson and SPIFFS.

Fig 5.11 Node MCU

NodeMCU was created shortly after the ESP8266 came out. On December 30, 2013, Espressif Systems began production of the ESP8266. The ESP8266 is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications. NodeMCU started on 13 Oct 2014, when Hong committed the first file of nodemcu-firmware to GitHub. Two months later, the project expanded to include an open-hardware platform when

developer Huang R committed the gerber file of an ESP8266 board, named devkit v0.9.

Later that month, Tuan PM ported MQTT client library from Contiki to the ESP8266 SoC platform, and committed to NodeMCU project, then NodeMCU was able to support the MQTT IoT protocol, using Lua to access the MQTT broker. Another important update was made on 30 Jan 2015, when Devsaurus ported the u8glib to NodeMCU project, enabling NodeMCU to easily drive LCD, Screen, OLED, even VGA displays.

In summer 2015 the creators abandoned the firmware project and a group of independent contributors took over. By summer 2016 the NodeMCU included more than 40 different modules. Due to resource constraints users need to select the modules relevant for their project and build a firmware tailored to their needs.

As Arduino.cc began developing new MCU boards based on non-AVR processors like the ARM/SAM MCU and used in the Arduino Due, they needed to modify the Arduino IDE so that it would be relatively easy to change the IDE to support alternate tool chains to allow Arduino C/C++ to be compiled for these new processors. They did this with the introduction of the Board Manager and the SAM Core. A "core" is the collection of software components required by the Board Manager and the Arduino IDE to compile an Arduino C/C++ source file for the target MCU's machine language. Some ESP8266 enthusiasts developed an Arduino core for the ESP8266 WiFi SoC, popularly called the "ESP8266 Core for the Arduino IDE". This has become a leading software development platform for the various ESP8266-based modules and development boards, including NodeMCUs.

5.9 ENERGIA IDE

Energia is an open-source electronics prototyping platform started by Robert Wessel's in January of 2012 with the goal to bring the Wiring and Arduino framework to the Texas Instruments MSP430 based Launch Pad. The Energia IDE is cross platform and supported on Mac OS, Windows, and Linux. Energia uses the mspgcc compiler by Peter Bigot and is based the Wiring and Arduino framework.

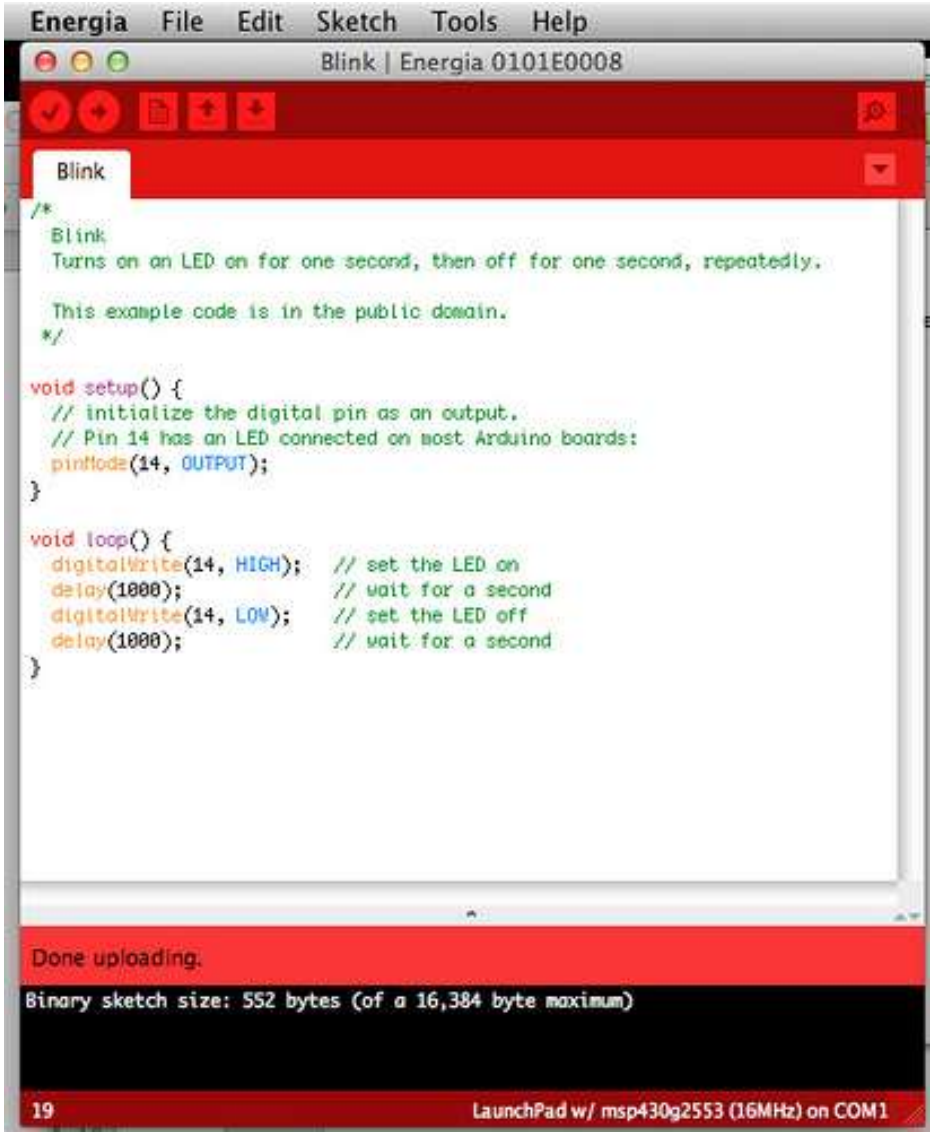

Fig 5.12 Energia Environment

Energia includes an integrated development environment (IDE) that has its foundation in the Processing IDE (Processing→Wiring→Arduino→Energia). Energia is also a portable framework/abstraction layer that can be used in other popular IDEs. Utilize a web browser based environment with Texas Instruments CCS Cloud at dev.ti.com or TI's power full CCS Desktop IDE. The foundation of Energia and Arduino is the Wiring framework that was developed by Hernando Barragan.

The framework is thoughtfully created with designers and artists in mind to encourage a community where both beginners and experts from around the world share ideas, knowledge and their collective experience. The Energia team adopts the philosophy of learning by doing and strives to make it easy to work directly with the hardware. Professional engineers, entrepreneurs, makers, and students can all benefit from the ease of use Energia brings to the microcontroller. Energia started out to bring the Wiring and Arduino framework to the Texas Instruments MSP430 launch Pad.

The launch Pad is a low-cost microcontroller board that is made by Texas Instruments. The latest release of Energia supports the majority of the launch Pad product offerings. Energia introduces a together with Energia , Launch Pad can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Launch Pad projects can be stand-alone (only run on the Target Board, i.e. your Launch Pad), or they can communicate with software running on your computer (Host PC). You can also add wireless modules to enable communication over various types of RF including Wi-Fi, NFC, Bluetooth, Zigbee, cellular, and more.

5.10 BLYNK API

Blynk is an Internet of Things (IoT) service designed to make remote control and reading sensor data from your devices as quick and easy as possible. In this

article we will cover exactly what Blynk is, how it works, and provide two short example projects on different uses of the service with NodeMCU and Raspberry Pi development boards.

One area which can pose a problem for the uninitiated is coding and networking. Blynk aims to remove the need for extensive coding, and make it easy to access your devices from anywhere on your smartphone. It's free to use for hobbyists and developers, though it's also available to use commercially for a fee — companies can use Blynk to create their own apps and systems then sell them with their own branding. Blynk uses its own server and library in order to make the service work, but it's the Blynk app that seems to be its main strength.

5.10.1 THE BLYNK APP

The Blynk app is available for free on Android and iOS. It's the starting point for your projects, featuring a simple to use drag and drop system for building custom controls for your IoT setup. The workflow is fast: when starting a new project you're prompted to choose your development board from an extensive list, and also your method of connection. The app then sends an authorization token via email for connecting to your device over the Blynk server.

Control elements are called Widgets: various types of input methods and output displays including buttons, sliders, a joystick, graphs and text feedback. There are also component specific widgets, with stylized controls for LEDS, LCD displays, and even live streamed video. Also notable are widgets that add features, like automatic posting to twitter, and custom notifications.

While the app is free, it limits how many widgets you can use at once by giving them all an "Energy" cost. The app gives you a balance of 2,000 to play

with, with the option to buy more if needed. I found that the starting balance provided was more than enough for the example projects listed here, though if your setup is more complicated you might find yourself running out of juice quite quickly.

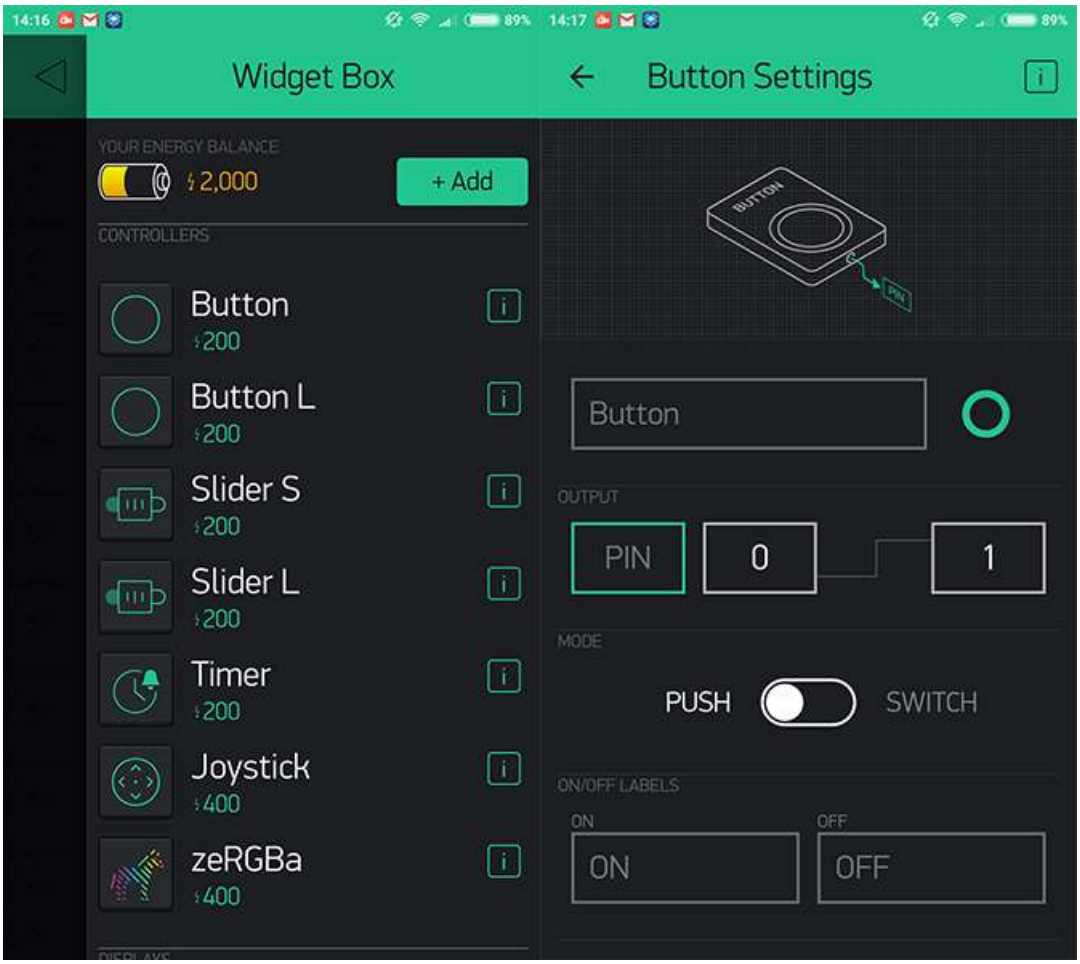

Fig 5.13 Blynk Interface

Each widget has an editing menu allowing you to change the name and colour. You then choose which pin to affect (whether it be a pin on your board or one of Blynk's virtual pins) along with the range of values to send. For output displays such as graphs and text boxes, you can also choose how often you wish it to be updated, potentially saving precious bandwidth. Blynk also features the ability to assign instructions to "virtual" pins, which are user configured connections between the app and the hardware. A single button in

the app can therefore be used to trigger many different events on the device. We'll cover how to use these later in the article.

The app gives the option of sharing your project with others. A QR code is generated which can be sent via email or scanned directly, and used by anyone who also has the Blynk app. Whoever you share with cannot make changes to the project, making it a quick and convenient way to share control of your devices. It is worth noting however that your project in the app must be running for others to have access to the hardware.

You may also share the project without allowing access to your hardware, which is a great way to teach people how to use the app without letting them turn your lights on and off! Once created, you can start using it immediately by pressing the play symbol in the top right corner. If you need to make changes later you can simply press the same button to go back into editing mode.

5.10.2 BLYNK SERVER

Once you have created an app to control your device, you have two options for how to communicate with it. The Blynk cloud server is quick, responsive, and free to use. Connecting to a Wi-Fi device is as easy as copying your generated authorization code into your Arduino sketch, and providing your Wi-Fi details. For Raspberry Pi, Blynk provide a test script which you can run with your authorization code to the same effect. Later in this article, we will create our own script using the Blynk library to connect to the service.

The second option is to host your own Blynk server. Blynk provide an open source Netty based Java server which can be run from your computer, or even a Raspberry Pi. This has various benefits for some users in terms of functionality and security, though for our examples here we will concentrate on using the Blynk cloud server provided.

5.10.3 BLYNK WITH ARDUINO IDE

Install the Blynk app on your smartphone, and create an account. Make sure you use an email address you can actually access as that is where your authorization tokens will be sent. Now create a project, selecting which board you will be using and how you will be connecting to it. Both examples here connect via Wi-Fi, though connections via Bluetooth, Ethernet, and even GSM are also possible.

Create your project. This will automatically send an authorization token. If you don’t receive it, you can resend it by selecting the project settings icon (the little nut), selecting your device, and selecting “E-mail”. Next, install the Blynk libraries from the Blynk website. For Arduino, install the library by copying the files into your Arduino > libraries folder.