DD SLAVE DEVICE

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1. INTRODUCTION:

A data logger (also datalogger or data recorder) is an electronic device that records data over time or in relation to location either with a built in instrument or sensor or via external instruments and sensors. One of the primary benefits of using data loggers is the ability to automatically collect data on a 24-hour basis. Upon activation, data loggers are typically deployed and left unattended to measure and record information for the duration of the monitoring period. This allows for a comprehensive, accurate picture of the environmental conditions being monitored, such as air temperature and relative humidity.

1.2. OBJECTIVES

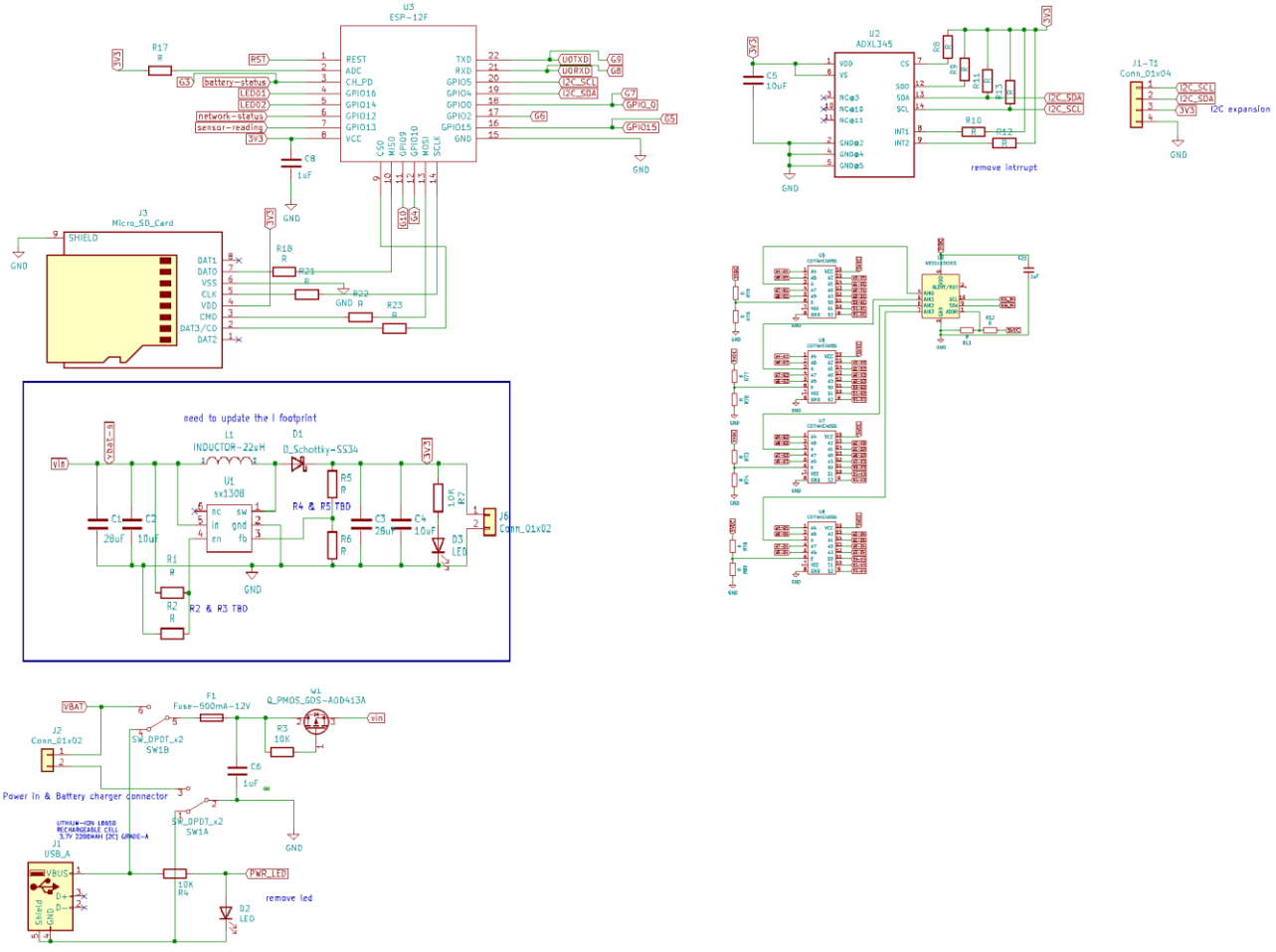
The major objectives of the proposed project are listed as follows:

- To acquist sensor data
- To store acquisted sensor data
- To send acquisted data to cloud
- To create an efficient way to mangae data in process industries

CHAPTER 2

2.1 CIRCUIT DIAGRAM OFDD SLAVES

The circuit diagram of DD SLAVE is shown below. It gives the complete circuit connections for DD SLAVE.



Circuit diagram for DD SLAVE

CHAPTER 3

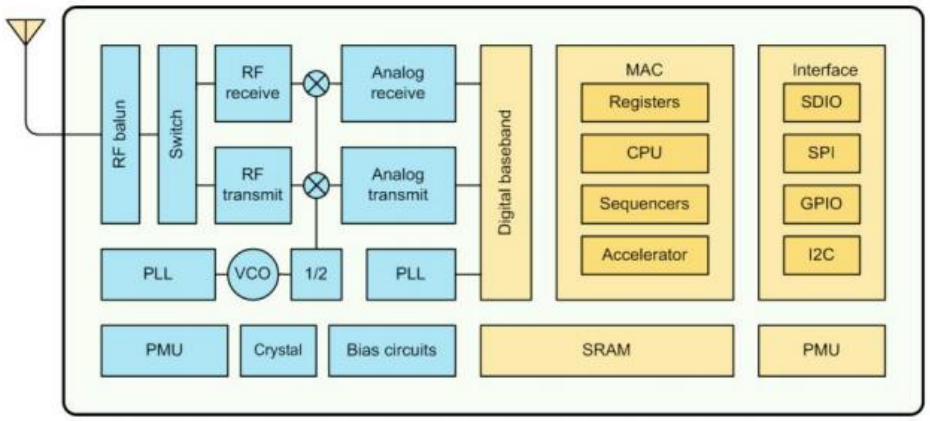
HARDWARE DESCRIPTION OF DD SLAVES

3.0 COMPONENTS USED FORDD SLAVES

In order to develop the hardware assembly of DD SLAVES**,** various components are to be used, they are discussed in the following sections

3.1 ESP 12E

ESP-12E WiFi module is developed by Ai-thinker Team. core processor ESP8266 in smaller sizes of the module encapsulates Tensilica L106 integrates industry-leading ultra low power 32-bit MCU micro, with the 16-bit short mode, Clock speed support 80 MHz, 160 MHz, supports the RTOS, integrated Wi-Fi MAC/BB/RF/PA/LNA, on-board antenna. The module supports standard IEEE802.11 b/g/n agreement, complete TCP/IP protocol stack. Users can use the add modules to an existing device networking, or building a separate network controller. ESP8266 is high integration wireless SOCs, designed for space and power constrained mobile platform designers. It provides unsurpassed ability to embed Wi-Fi capabilities within other systems, or to function as a standalone application, with the lowest cost, and minimal space requirement.



ESP8266EX Block Diagram

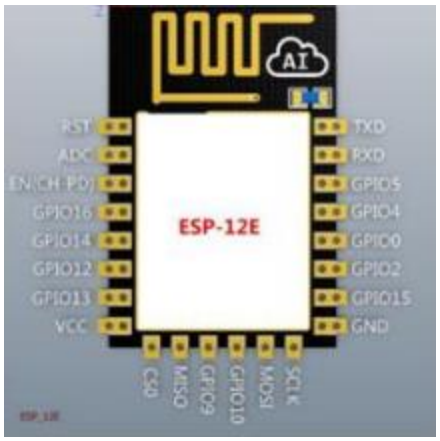
ESP8266EX also integrates an enhanced version of Tensilica's L106 Diamond series 32-bit processor, with on-chip SRAM, besides the Wi-Fi functionalities. ESP8266EX is often integrated with external sensors and other application specific devices through its GPIOs; codes for such applications are provided in examples in the SDK. Espressif Systems' Smart Connectivity Platform (ESCP) demonstrates sophisticated system-level features include fast sleep/wake context switching for energy-efficient VoIP, adaptive radio biasing. for low-power operation, advance signal processing, and spur cancellation and radio co-existence features for common cellular, Bluetooth, DDR, LVDS, LCD interference mitigation

Parameters

Categories	Items	Values
WiFi Paramters	WiFi Protocles	802.11 b/g/n
	Frequency Range	2.4GHz-2.5GHz (2400M-2483.5M)
Hardware Paramaters	Peripheral Bus	UART/HSPI/I2C/I2S/Ir Remote Contorl
		GPIO/PWM
	Operating Voltage	3.0~3.6V
	Operating Current	Average value: 80mA
	Operating Temperature Range	-40°~125°
	Ambient Temperature Range	Normal temperature
	Package Size	16mm*24mm*3mm
	External Interface	N/A
Software Parameters	Wi-Fi mode	station/softAP/SoftAP+station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network) / download and write firmware via host
	Ssoftware Development	Supports Cloud Server Development / SDK for custom firmware development
	Network Protocols	IPv4, TCP/UDP/HTTP/FTP
	User Configuration	AT Instruction Set, Cloud Server, Android/iOS App

Pin Descriptions

ESP-12E Pin design



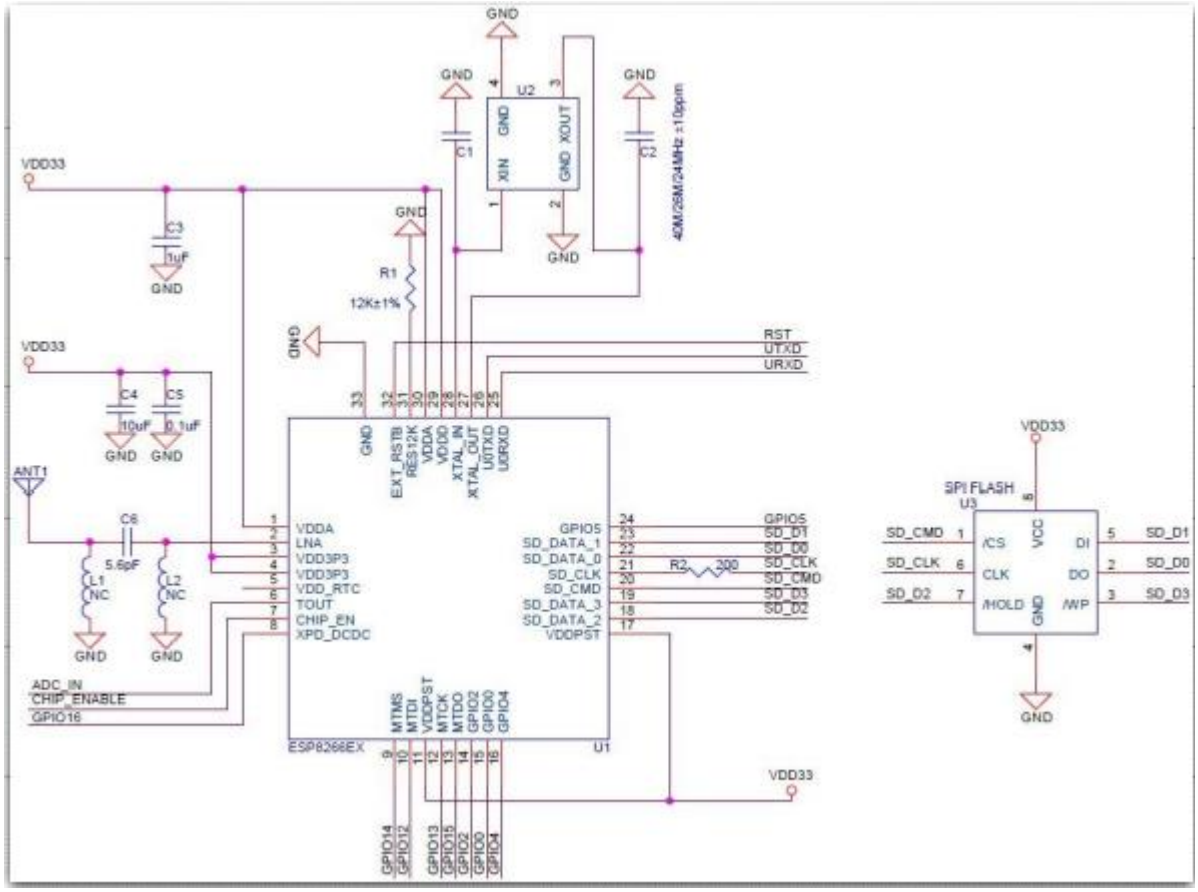
NO.	Pin Name	Function
1	RST	Reset the module
2	ADC	A/D Conversion result.Input voltage range 0-1v,scope:0-1024
3	EN	Chip enable pin.Active high
4	IO16	GPIO16; can be used to wake up the chipset from deep sleep mode.
5	IO14	GPIO14; HSPI_CLK
6	IO12	GPIO12; HSPI_MISO
7	IO13	GPIO13; HSPI_MOSI; UART0_CTS
8	VCC	3.3V power supply (VDD)
9	CS0	Chip selection
10	MISO	Salve output Main input
11	IO9	GPIO9
12	IO10	GBIO10
13	MOSI	Main output slave input
14	SCLK	Clock
15	GND	GND
16	IO15	GPIO15; MTDO; HSPICS; UART0_RTS
17	IO2	GPIO2; UART1_TXD
18	IO0	GPIO0
19	IO4	GPIO4
20	IO5	GPIO5
21	RXD	UART0_RXD; GPIO3
22	TXD	UART0_TXD; GPIO1

Interfaces

Descriptions of Interfaces

Interface	Pin Name	Description
HSPI	IO12(MISO) IO13(MOSI) IO14(CLK) IO15(CS)	SPI Flash 2, display screen, and MCU can be connected using HSPI interface.
PWM	IO12(R) IO15(G) IO13(B)	Currently the PWM interface has four channels, but users can extend the channels according to their own needs. PWM interface can be used to control LED lights, buzzers, relays, electronic machines, and so on.
IR Remote Control	IO14(IR_T) IO5(IR_R)	The functionality of Infrared remote control interface can be implemented via software programming. NEC coding, modulation, and demodulation are used by this interface. The frequency of modulated carrier signal is 38KHz.
ADC	TOUT	ESP8266EX integrates a 10-bit analog ADC. It can be used to test the power-supply voltage of VDD3P3 (Pin3 and Pin4) and the input power voltage of TOUT (Pin 6). However, these two functions cannot be used simultaneously. This interface is typically used in sensor products.
I2C	IO14(SCL) IO2(SDA)	I2C interface can be used to connect external sensor products and display screens, etc.
UART	**UART0:** TXD (U0TXD) RXD (U0RXD) IO15 (RTS) IO13 (CTS) **UART1:** IO2(TXD)	Devices with UART interfaces can be connected with the module. Downloading: U0TXD+U0RXD or GPIO2+U0RXD Communicating: UART0: U0TXD, U0RXD, MTDO (U0RTS), MTCK (U0CTS) Debugging: UART1_TXD (GPIO2) can be used to print debugging information. By default, UART0 will output some printed information when the device is powered on and is booting up. If this issue exerts influence on some specific applications, users can exchange the inner pins of UART when initializing, that is to say, exchange U0TXD, U0RXD with U0RTS, U0CTS.
I2S	**I2S Input:** IO12 (I2SI_DATA) ; IO13 (I2SI_BCK); IO14 (I2SI_WS); **I2S Output:** IO15 (I2SO_BCK); IO3 (I2SO_DATA); IO2 (I2SO_WS).	I2S interface is mainly used for collecting, processing, and transmission of audio data.

Shecmatic



Schematics of Esp-12E WiFi Module

3.2 USB TO TTL PROGRAMING CIRCUIT

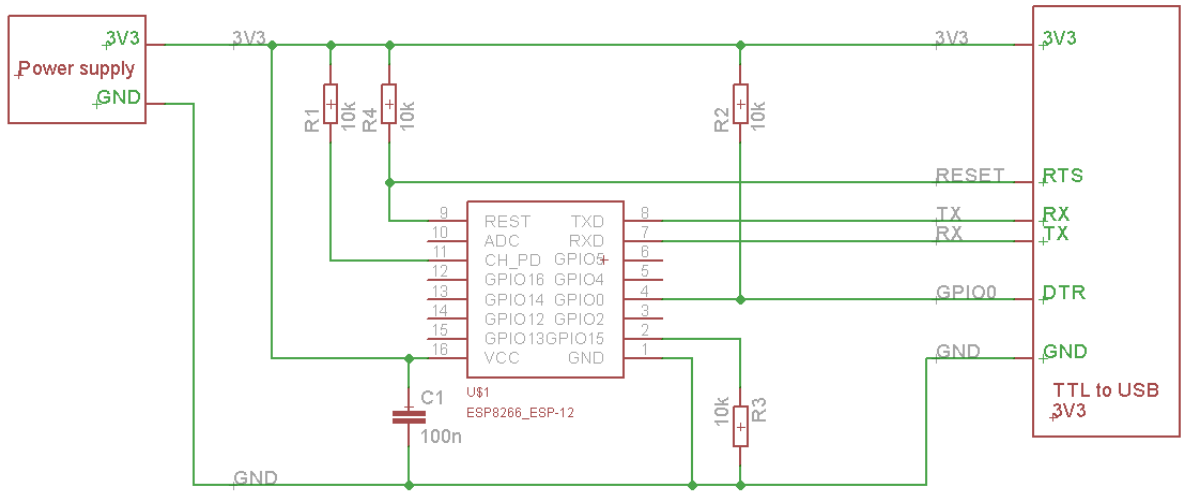
The behaviour described above happens thanks to a special piece of code that is executed at every reset of the microcontroller and that looks for a sketch to be uploaded from the serial/USB port using a specific protocol and speed. If no connection is detected, the execution is passed to the code of your sketch.This little (usually 512 bytes) piece of code is called the “Bootloader” and it is in an area of the memory of the microcontroller – at the end of the address space - that can’t be reprogrammed as a regular sketch and had been designed for such purpose

Minimal Hardware Setup for Bootloading and Usage:

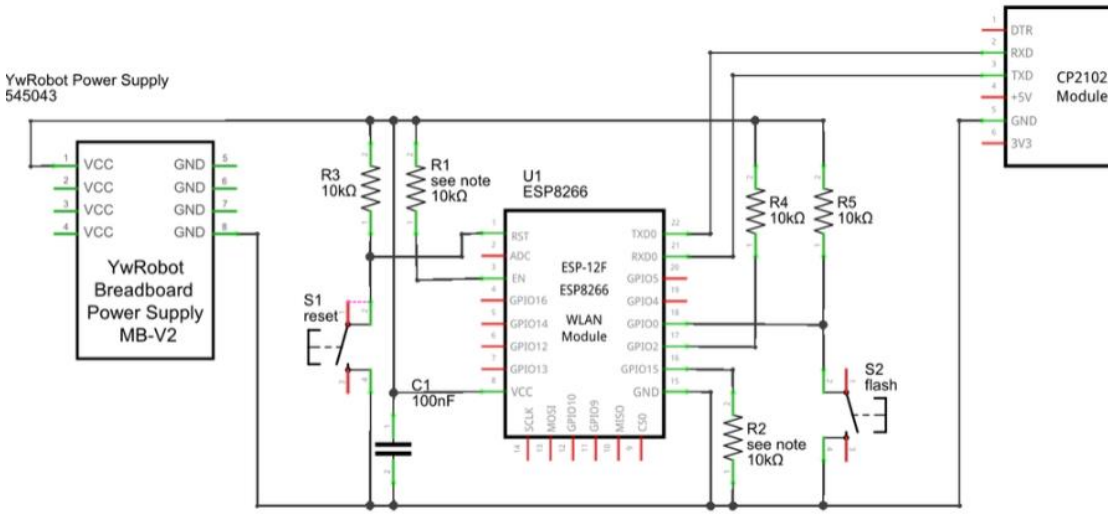
PIN	Resistor	Serial Adapter
VCC		VCC (3.3V)
GND		GND
TX or GPIO2*		RX
RX		TX
GPIO0	PullUp	DTR
Reset*	PullUp	RTS
GPIO15*	PullDown	
CH_PD	PullUp	

Note

- GPIO15 is also named MTDO
- Reset is also named RSBT or REST (adding PullUp improves the stability of the module)
- GPIO2 is alternative TX for the boot loader mode
- **Directly connecting a pin to VCC or GND is not a substitute for a PullUp or PullDown resistor, doing this can break upload management and the serial console, instability has also been noted in some cases.**



Minimal Circuit done :



- EN (CH_PD) Enable Pin has to be pulled up (R1)
- RESET Pin has to be pulled up (R3). In order to retart the module we connect also a push button to ground (S1)
- In all of the boot modes
    - GIO15 has to be pulled down (R2)
    - GIPO2 has to be pulled up (R4)
- GIPO0 has to be pulled up (R5) for running the program (flash boot). I has to be low in order to enter programming mode (UART). For this we add the S2 push button.
- USB to Serial port
    - GND are connected together
    - RX module is connected to TX serial
    - TX module is connected to RX serial

FLASH PROGRAMMING

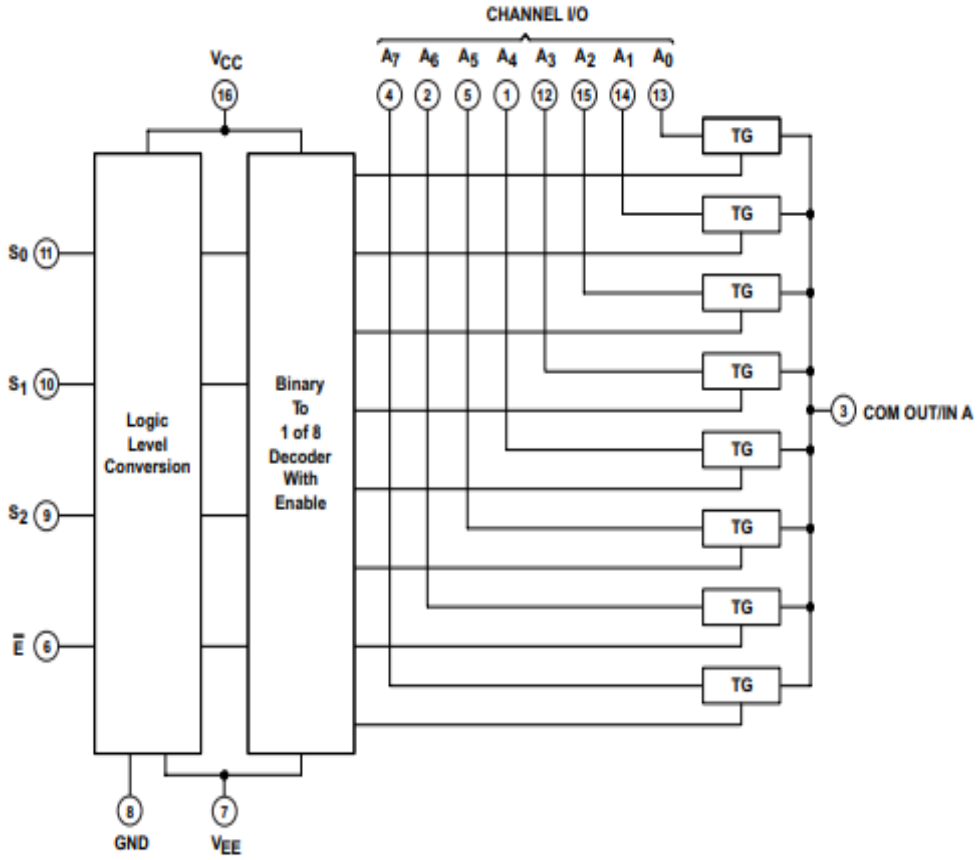
- To Upload program in Arduino IDE.
- Keep pressed the FLASH button.
- When "esptool.exe .... -cp COM...." appears in the messages area then quickly press and release RESET. Then release FLASH button.
- Check for the messages area the flashing process.
- After the flashing the you should see the the blue led blinking.

3.3 MULTIPELXER CIRCUIT

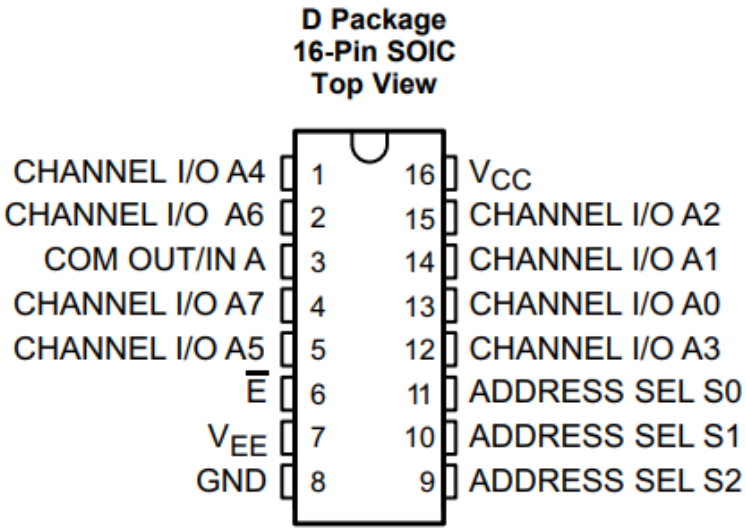
The 74HC4051 allows you to turn four I/O pins into eight multi-functional, individually-selectable signals, which can be used do everything from driving eight LEDs to monitoring eight potentiometers. A multiplexer, commonly abbreviated down to **"mux"**, is an electronically-actuated switch, which can turn one signal into many. It routes a common input signal to any number of separate outputs. Similarly, a *de*multiplexer routes any number of selectable inputs to a single common output.

The 74HC4051 can function as either a multiplexer or a demultiplexer, and it features **eight channels** of selectable inputs/outputs. The routing of common signal to independent I/O is set by digitally controlling three **select lines**, which can be set either high or low into one of eight binary combinations.

Functional Block Diagram



Pin Configuration And Functions



Pin Functions

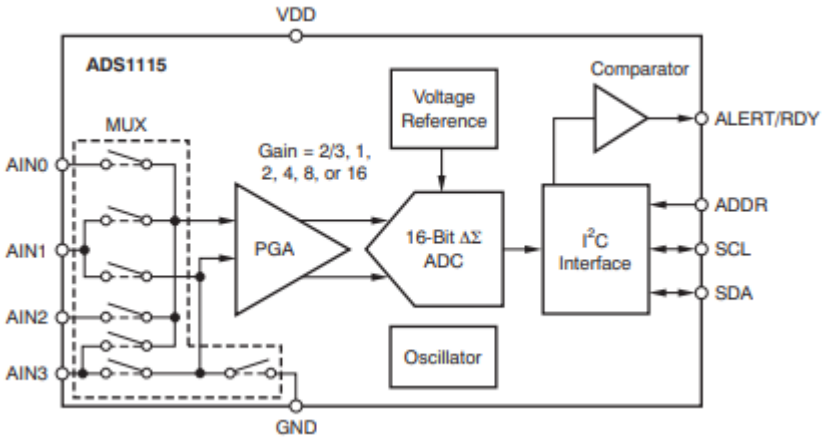
PIN		I/O	DESCRIPTION
NAME	NO.		
A4	1	I/O	Channel 4 input / output
A6	2	I/O	Channel 6 Input / output
A	3	I/O	COM OUT/ IN
A7	4	I/O	Channel 7 Input / Output
A5	5	I/O	Channel 5 Input / Output
Ebar	6	I	Enable input
VEE	7	I	Power input level for incoming Channel
GND	8	I	Power GND
VCC	9	I	Power input level for outgoing Channel
A2	10	I/O	Channel 2 Input / Output
A1	11	I/O	Channel 1 Input / Output
A0	12	I/O	Channel 0 Input / Output
A3	13	I/O	Channel 3 Input / Output
S0	14	I	Address Select Input 0
S1	15	I	Address Select Input 1
S2	15	I	Address Select Input 2

74HC4051 Logic Table

$\bar{E}$	S2	S1	S0	I/O Connected to Z
L	L	L	L	Y0
L	L	L	H	Y1
L	L	H	L	Y2
L	L	H	H	Y3
L	H	L	L	Y4
L	H	L	H	Y5
L	H	H	L	Y6
L	H	H	H	Y7
H	X	X	X	None

3.4 ADC CIRCUIT

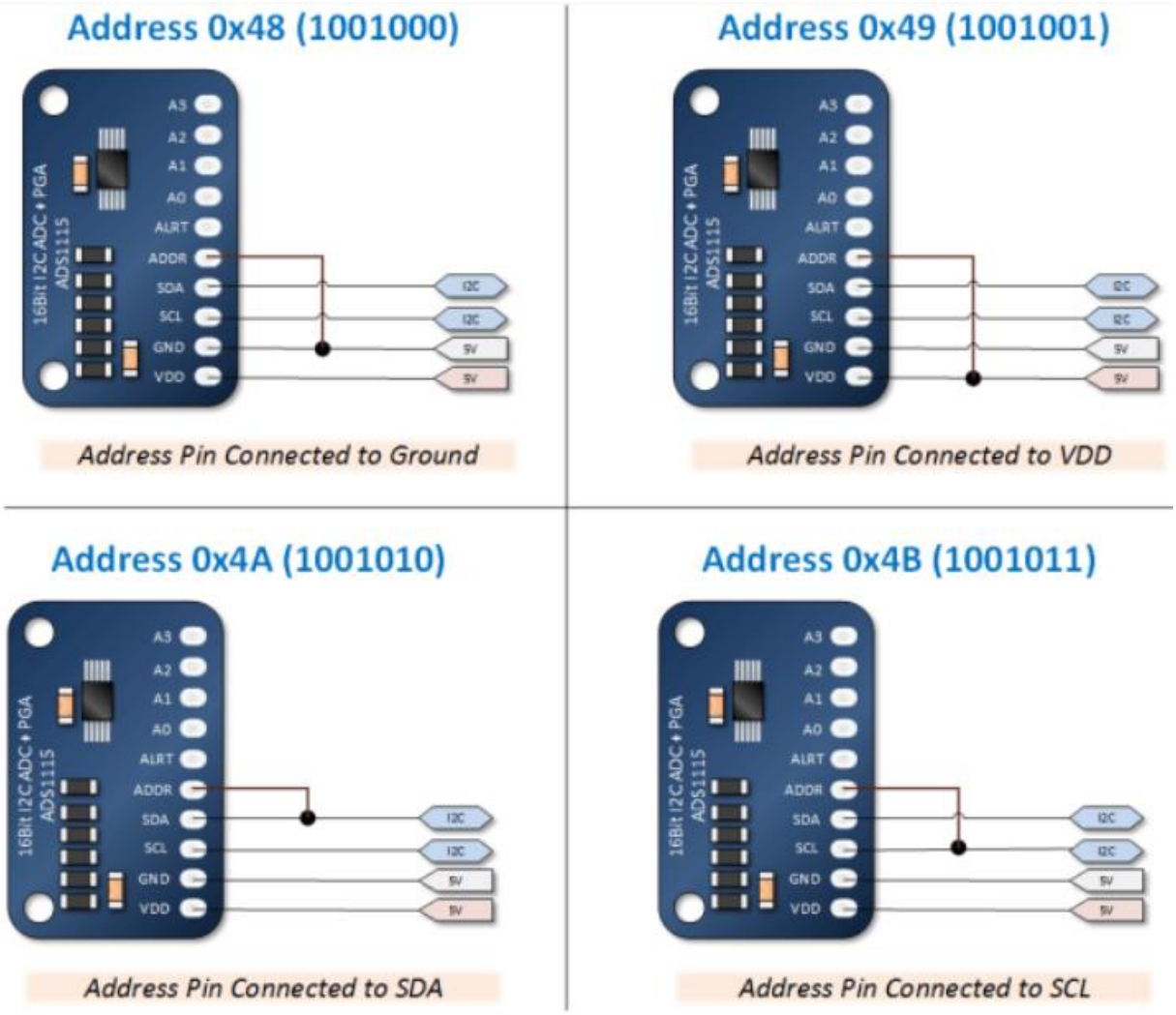
The ADS1113/4/5 are very small, low-power, 16-bit, followed by a digital filter. Input signals are compared delta-sigma (ΔΣ) analog-to-digital converters (ADCs). to the internal voltage reference. The digital filter The ADS1113/4/5 are extremely easy to configure receives a high-speed bitstream from the modulator and design into a wide variety of applications, and and outputs a code proportional to the input voltage. allow precise measurements to be obtained with very little effort. Both experienced and novice users of The ADS1113/4/5 have two available conversion data converters find designing with the ADS1113/4/5 modes: single-shot mode and continuous conversion family to be intuitive and problem-free. mode. In single-shot mode, the ADC performs one conversion of the input signal upon request and The ADS1113/4/5 consist of a ΔΣ analog-to-digital stores the value to an internal result register. The (A/D) core with adjustable gain (excludes the device then enters a low-power shutdown mode. This ADS1113), an internal voltage reference, a clock mode is intended to provide significant power savings oscillator, and an I 2C interface. An additional feature in systems that only require periodic conversions or available on the ADS1114/5 is a programmable digital when there are long idle periods between comparator that provides an alert on a dedicated pin. conversions. In continuous conversion mode, the All of these features are intended to reduce required ADC automatically begins a conversion of the input external circuitry and improve performance. signal as soon as the previous conversion is shows the ADS1115 functional block diagram. completed. The rate of continuous conversion is The ADS1113/4/5 A/D core measures a differential equal to the programmed data rate. Data can be read signal, V at any time and always reflect the most recent IN, that is the difference of AINP and AINN. A MUX is available on the ADS1115. This architecture completed conversion.



ADS1115 Functional Block Diagram

I 2C ADDRESS SELECTION

The ADS1113/4/5 can act as either slave receivers or The ADS1113/4/5 have one address pin, ADDR, that slave transmitters. As a slave device, the sets the I 2C address. This pin can be connected to ADS1113/4/5 cannot drive the SCL line. ground, VDD, SDA, or SCL, allowing four addresses to be selected with one pin. The Receive Mode: state of the address pin ADDR is sampled In slave receive mode the first byte transmitted from continuously. the master to the slave is the address with the R/W bit low. This byte allows the slave to be written. ADDR Pin Connection and The next byte transmitted by the master is the Corresponding Slave Address register pointer byte. The ADS1113/4/5 then ADDR PIN SLAVE ADDRESS acknowledge receipt of the register pointer byte. The Ground 1001000 next two bytes are written to the address given by the VDD 1001001 register pointer. The ADS1113/4/5 acknowledge each byte sent. Register bytes are sent with the most SDA 1001010 significant byte first, followed by the least significant SCL 1001011 byte.



I 2C GENERAL CALL Transmit Mode

The ADS1113/4/5 respond to the I 2C general call In slave transmit mode, the first byte transmitted by address (0000000) if the eighth bit is '0'. The devices the master is the 7-bit slave address followed by the acknowledge the general call address and respond to high R/W bit. This byte places the slave into transmit commands in the second byte. If the second byte is mode and indicates that the ADS1113/4/5 are being 00000110 (06h), the ADS1113/4/5 reset the internal read from. The next byte transmitted by the slave is registers and enter power-down mode. the most significant byte of the register that is indicated by the register pointer. This byte is followed I by an acknowledgment from the master. The I2C SPEED MODES remaining least significant byte is then sent by the The I 2C bus operates at one of three speeds. slave and is followed by an acknowledgment from the Standard mode allows a clock frequency of up to master. The master may

terminate transmission after 100kHz; fast mode permits a clock frequency of up to any byte by not acknowledging or issuing a START or 400kHz; and high-speed mode (also called Hs mode) STOP condition. allows a clock frequency of up to 3.4MHz. The ADS1113/4/5 are fully compatible with all three WRITING/READING THE REGISTERS modes. To access a specific register from the ADS1113/4/5, No special action is required to use the ADS1113/4/5 the master must first write an appropriate value to the in standard or fast mode, but high-speed mode must Pointer register. The Pointer register is written directly be activated. To activate high-speed mode, send a after the slave address byte, low R/W bit, and a special address byte of 00001xxx following the successful slave acknowledgment. After the Pointer START condition, where xxx are bits unique to the register is written, the slave acknowledges and the Hs-capable master. This byte is called the Hs master master issues a STOP or a repeated START code.

3.5 ACCELEROMETER CIRCUIT:

The ADXL345 is a small, thin, low power, 3-axis MEMS accelerometer with high resolution (13-bit) measurement at up to +/-16 g. Digital output data is formatted as 16-bit two's complement and is accessible through either an SPI (3- or 4-wire) or I^2C digital interface.

Hardware Overview

Features

- Supply Voltage: 2.0 - 3.6 VDC
- Ultra Low Power: As low as 23 uA in measurement mode, 0.1uA in standby mode at 2.5V
- SPI or I2C Communication
- Single Tap / Double Tap Detection
- Activity / Inactivity Sensing
- Free-Fall Detection

The single and double tap sensing detects when a single, or two simultaneous, acceleration events occur. Activity and inactivity sensing detect the presence or lack of motion. Free-fall sensing compares the acceleration on all axes with the threshold value to know if the device is falling. All thresholds levels that trigger the activity, free-fall, and single tap/double tap events are *user-set* levels. These functions can also be mapped to one of two interrupt output pins. An integrated, patent pending 32-level first in, first out (FIFO) buffer can be used to store data to minimize host processor intervention.

The ADXL345 is well suited to measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (4 mg/LSB) enables measurement of inclination changes less than 1.0°. Furthermore, low power modes enable intelligent motion-based power management with threshold sensing and active acceleration measurement at extremely low power dissipation.

Pin Function Descriptions

Mnemonic	Description
GND	This pin must be connected to ground
VCC	Supply Voltage
CS	Chip Select
INT1	Interrupt 1 Output
INT2	Interrupt 2 Output
SDO	Serial Data Output (SPI 4-Wire) / I2C Address Select
SDA / SDI / SDIO	Serial Data I2C / Serial Data Input (SPI 4-WIRE) / Serial Data Input and Output (SPI 3-Wire)
SCL/SCLK	Serial Communications Clock

I2C Communication:

I2C mode is enabled if the CS pin is tied to high. There is no default mode if the CS pin is left unconnected, so it should always be tied high or driven by an external controller.

Arduino Pin	ADXL345 Pin
GND	GND
3V3	VCC
3V3	CS
GND	SDO
A4	SDA
A5	SCL

Program :

```
#include <Wire.h>
#include <Adafruit_ADS1015.h>
Adafruit_ADS1115 ads(0x49);
float Voltage = 0.0;
const int selectPins[3] = {10, 9, 8}; // S0~8, S1~9, S2~10
const int zOutput = 5;
int time = 1;
int Signal=0;
const int zInput = A0; // Connect common (Z) to A0 (analog input)
void setup()
{
  Serial.begin(9600); // Initialize the serial port
  ads.begin();
  for (int i=0; i<3; i++)
  {
    pinMode(selectPins[i], OUTPUT);
    digitalWrite(selectPins[i], HIGH);
  }
  pinMode(zInput, INPUT); // Set up Z as an input
}
void loop()
{
  for (int n=0;n<8;n++)
  {
    if(time<=10){
      digitalWrite(selectPins[0], LOW);
      digitalWrite(selectPins[1], LOW);
      digitalWrite(selectPins[2], LOW);
```

```
  Signal=1;
 }
 else if(time<=20&&time>10) {
  digitalWrite(selectPins[0], LOW);
  digitalWrite(selectPins[1], LOW);
  digitalWrite(selectPins[2], HIGH);
  Signal=2;
  }
 else if(time<=30&&time>20) {
  digitalWrite(selectPins[0], LOW);
  digitalWrite(selectPins[1], HIGH);
  digitalWrite(selectPins[2], LOW);
  Signal=3;
  }
 else if(time<=40&&time>30) {
  digitalWrite(selectPins[0], LOW);
  digitalWrite(selectPins[1], HIGH);
  digitalWrite(selectPins[2], HIGH);
  Signal=4;
  }
  else if(time<=50&&time>40) {
  digitalWrite(selectPins[0], HIGH);
  digitalWrite(selectPins[1], LOW);
  digitalWrite(selectPins[2], LOW);
  Signal=5;
  }
  else if(time<=60&&time>50) {
  digitalWrite(selectPins[0], HIGH);
  digitalWrite(selectPins[1], LOW);
```

```
    digitalWrite(selectPins[2], HIGH);
    Signal=6;
    }
    else if(time<=70&&time>60) {
    digitalWrite(selectPins[0], HIGH);
    digitalWrite(selectPins[1], HIGH);
    digitalWrite(selectPins[2], LOW);
    Signal=7;
    }
    else {
    digitalWrite(selectPins[0], HIGH);
    digitalWrite(selectPins[1], HIGH);
    digitalWrite(selectPins[2], HIGH);
    Signal=8;
    }

int inputValue = analogRead(A0); // and read Z
int16_t adc0; // we read from the ADC, we have a sixteen bit integer as a result
adc0 = ads.readADC_SingleEnded(0);
Voltage = (adc0 * 0.1875)/1000;
Serial.print("Time:"+String(time)+"\t"+"\t");
Serial.print("Signal:"+String(Signal)+"\t");
Serial.print("AIN0: "+ String(adc0) +"\t");
Serial.print("Analog"+String(inputValue)+"\t" );
Serial.print("Voltage: "+String(Voltage)+"\t");
Serial.print("\n");
delay(1000);
time=time+1;
if(time==81)
```

```
  {
    time=0;
    Signal=0;
  }}
}
```

REFERENCES

1. ESP 12E : https://en.wikipedia.org/wiki/ESP8266
2. MULTIPLEXER: https://learn.sparkfun.com/tutorials/multiplexer-breakout-hookup-guide/all
3. ADS115: https://www.instructables.com/id/16-bit-I2C-Temperature-Monitor-Using-Arduino/
4. ADXL345: https://learn.sparkfun.com/tutorials/adxl345-hookup-guide/all
5. USB TO TLL: https://www.instructables.com/id/ESP-12F-ESP8266-Module-Minimal-Breadboard-for-Flas/ and http://arduino.esp8266.com/Arduino/versions/2.3.0/doc/boards.html