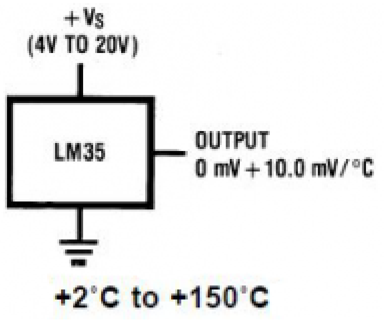
Lab Work 3 - Peripheral Interfacing

This module covers working on interfacing the PI with commonly used peripherals.

Temperature Sensor

One of the most commonly used sensor is the temperature sensor. There are quite a number of sensors to choose from and it all depends on your requirement.

LM35 Precision Centigrade Temperature Sensor



If cost is your primary concern, then this would be your first choice. This precision integrated-circuit temperature sensor has an output voltage that is linearly proportional to temperature in centigrade. However, the fact that it provides a range of voltage level as output requires an interface circuit (ADC) to be used.

The LM35 requires a 5V supply - not ideal, but we can still power it from Pi's 5V pin. It has just 3 pins, 2 for the power supply and one for the analog output. The output pin provides an analog voltage output that is linearly proportional to temperature in Celcius. Measurement resolution is 1 millivolt per 0.1°C (10mV per degree). So, an output voltage of 315mV (0.315V) is equivalent to a temperature of 31.5°C.

The Raspberry Pi is designed to be a low-cost computer and therefore, does not require any analog input. So, to actually use this sensor with Pi, we need an ADC - a 10-bit ADC would be recommended.

Datasheet here.

DS1620 Digital Thermometer and Termostat

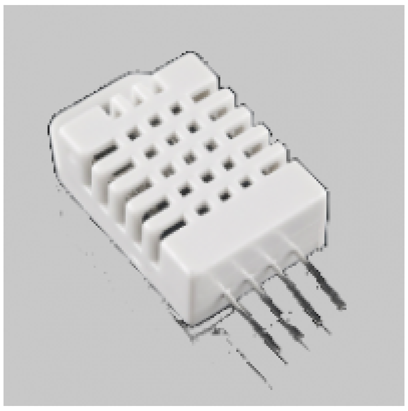
Having a digital solution is always preferable when designing an embedded system. It makes things much easier to interface. However, some will require certain protocols, like this one

The DS1620 requires 3-wire communications protocol RST_ (active low signal), CLK and DQ (valid on CLK hi). It also needs a developer to read up some stuffs about configuration/status registers and the command to request for temperature.

Datasheet here.

AM2302/DHT22 Digital Relative Humidity and Temperature Sensor

This is what we will be using at the lab.



Note: The picture above correctly shows a 4-pin component. The pins that actually have to be connected are only three (1 unused) - the ones that we have at the lab come with 'breakout' boards that pulls out only three pins.

This sensor can be directly connected to our Pi. It can be powered with 3.3V supply and the output is a single-wire digital line. This would be a good example to show how a GPIO pin can be configured as input AND output to facilitate bidirectional transfer.

Sample source code to interface this sensor to Pi is available here.

Datasheet here.

GSM Module

This is the module of choice when some kind of communications is needed by an embedded system. The basic ones can make voice calls or at least send an SMS (used to be the favorite choice for alert method… not much so since 3G/4G services becoming cheaper). The more advanced once comes with a TCP stack - that's networking protocol, baby! So, we will look into some (one?) of them - most probably the cheap one(s) that only provide call/SMS

Siemens TC35

This board is what we have at the lab.

Although the board can be powered from a 3.3V voltage source (like the 3.3V pin on Raspberry Pi!), the supply pin on Raspberry Pi fails to provide enough current for the GSM module to operate with the SIM card inserted (testing without the SIM card is not a problem). This is because when the SIM card is inserted, it drains up to 80-90mA in order to obtain a carrier signal. From here, looking at the sub-section on power pins, *“The maximum permitted current draw from the 3.3 V pins is 50 mA.”*. However, the next statement says, *“Maximum permitted current draw from the 5 V pin is the USB input current (usually 1 A) minus any current draw from the rest of the board.”*, which is great in our case because we are using a 2A USB power adapter. So, we will use the 5V pin from the Raspberry Pi and connect it to the 3.3V on the TC35 GSM module board. This is still okay because the datasheet of TC35 module (section 6.4 - power supply ratings) says that the maximum allowed voltage supply is 5.5V.

To connect to the board from a controller board, connect the serial transmit pin (TX) on your controller board to the TX0 pin on the GSM module board and the serial receive pin (RX) on your controller board to the RX0 pin on the GSM module board (as mentioned in this YouTube video, although I would not say it is wrong… it could be just that they're trying to fit into the DCE/DTE[1] thingy. But, what do I know?).

It is recommended to take out the MAX232 IC from the board since it will draw some current even when NOT in use. The only other thing left to consider is the start button - which requires human operator, and NOT acceptable in a 'smart' system. So, we need to replace that button with an interface pin that can be connected to a controller I/O pin.

Sample source code to interface this module to Pi is available here.

Datasheet here.

AT Commands

Mostly done using AT commands through serial communications (modem interface).

- Common AT commands for SIM908/900 module

The basic few commands that are necessary to know:

Command	Description
AT	Should return OK
A/	Re-issue the last command
ATE0@ATE	Disable local echo
ATE1	Enable local echo

Basic communication

To make voice call:

```
ATD01199441100;

// to check status
AT+CLCC

ATH
```

To make data call (no longer supported by provider?), just lose the semi-colon. The modem responds with `CONNECT: BBBB` (BBBB represents the baudrate).

To send SMS:

```
AT+CGMF=1

AT+CGMS="+601199441100"

// wait for '>' prompt

// send message and end with ctrl-z (0x1a @ 26), to cancel use ESC (0x1b @
27)
```

Module Info

Command	Description
AT+CGMI	Get manufacturer info
AT+CGMM	Get manufacturer model

Command	Description
AT+CGSN	Get IMEI (unique worldwide)
AT+CPIN?	Get SIM card status
AT+CSQ	Get signal quality

Other Things

To send USSD code e.g. *100#…

```
AT+CUSD=1,"<code>",15
```

To check own number

```
AT+CNUM
```

To register on network:

```
AT+CREG=? => register gsm (1 - no loc info, 2 - with loc info)

// Sample output:
+CREG: 2,5,"0FCE","AFF2"

// Indicator:
mode = 2
status = 5
  > 0 - not reg, not looking
  > 1 - reg, home
  > 2 - not reg, looking
  > 3 - reg denied
  > 4 - unknown
  > 5 - reg, roaming
lac = 0FCE (area)
ci = AFF2 (cell id)
```

To use data line:

```
at+cgatt=1 => attach gprs service

#at+cgdcont=1,"IP","<apn>" => define pdp context

at+cstt="<apn>","<user>","<pass>" => supply password?

at+ciicr => wireless with gprs???

at+cifsr => get location ip?

at+cipstatus => get gprs status

at+cipstart="tcp@udf?","<hostip@hostname>","<port>" => should get "CONNECT
OK" message
```

```
at+cipsend => start sending (ends with ctrl+z), should get "SEND OK" message

at+cipclose => close started connection

at+cipshut => shuts ciicr?
```

Dumped for now

```
* device mode ( e.g. 0 - data, 8 - voice )
=> check possible settings
at+fclass=?
+FCLASS: (0,1,2)
OK
=> check current settings
at+fclass?
+FCLASS: 0
OK

* check baud rate
at+ipr=?
+IPR: (0,2400,4800,9600,19200,38400,57600),(300,600,1200,115200)
OK
at+ipr?
+IPR: 115200
OK

* phone functionality ( 0 - minimum, 1 - full )
at+cfun=?
+CFUN: (0,1),(0,1)
OK
at+cfun?
+CFUN: 1
OK

* check indicators
+cind
at+cind=?
+CIND:
("battchg",(0-5)),("signal",(0-5)),("service",(0-1)),("message",(0-1)),("cal
l",(0-1)),("roam",(0-1)),("smsfull",(0-2))
OK
at+cind?
+CIND: 4,3,1,0,0,1,0
OK
```

Bluetooth Module

Very useful for low bandwidth / control-related applications. Usually, interfaced using serial UART terminal.

HC-06 (Slave-Only)

A slave only bluetooth module. Cannot initiate pairing.

AT Commands

Note that the AT commands here do not need a terminating character (i.e. 0x0d and/or 0x0a). The following are for HC-06 only!

AT Command	REply from HC-06	COMMENTs
AT	OK	Used to verify communication
AT+VERSION	OKlinvorV1.8	The firmware version
AT+NAMEmyBT	OKsetname	Sets the module name to 'myBT'
AT+PIN1234	OKsetPIN	Sets the module PIN to 1234
AT+BAUD1	OK1200	Sets the baud rate to 1200
AT+BAUD2	OK2400	Sets the baud rate to 2400
AT+BAUD3	OK4800	Sets the baud rate to 4800
AT+BAUD4	OK9600	Sets the baud rate to 9600
AT+BAUD5	OK19200	Sets the baud rate to 19200
AT+BAUD6	OK38400	Sets the baud rate to 38400
AT+BAUD7	OK57600	Sets the baud rate to 57600
AT+BAUD8	OK115200	Sets the baud rate to 115200

[1] You directly connect TX/RX (using straight cable) when connecting DCE to DTE, but you switch TX/RX (using null modem cable) when connecting DCE-DCE or DTE-DTE. Since a controller is a DTE (Data Terminal Equipment, by the way) and the GSM module is a DCE (that is, Data Communications@Carrier Equipment), it should use a straight cable connection.