Mobile Computing And IOT

Category	
Files	
Created	@September 20, 2022 8:14 PM
Reminder	
Status	Open
URL	
Updated	@December 15, 2022 1:21 PM

▼ unit 1

cellular arch -

cell into - https://www.youtube.com/watch?v=3jtEZDycKOQ&ab_channel=EzEdChannel

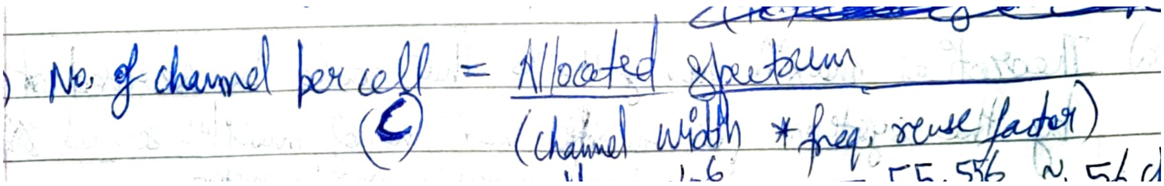
cell spilling - https://www.youtube.com/watch?v=7rjmKgMrDpM&ab_channel=ECITEngineering

same upar wala topic hindi - https://www.youtube.com/watch?v=X7qQq1Cb1fY&t=432s&ab_channel=EasyEngineeringClasses

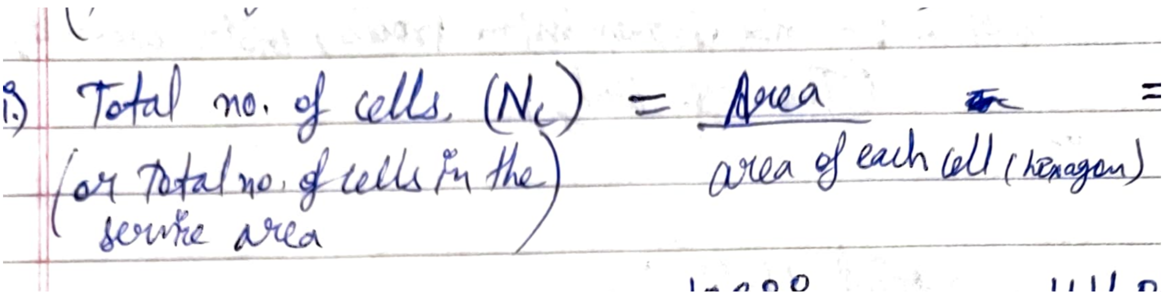
reuse,, handoff - https://www.youtube.com/watch?v=iojj6F1v59yw&ab_channel=EzEdChannel

practice sheet answers -

1-


No. of channel per cell (C) = $\frac{\text{Allocated Spectrum}}{(\text{channel width} * \text{freq. reuse factor})}$

intensity of eash cell we calculate from erlang chart


3) Total no. of cells (N_c) (or Total no. of cells in the service area) = $\frac{\text{Area}}{\text{area of each cell (hexagon)}}$

area of heaxagon -

$$\approx 2.5981$$

$$= \frac{10000}{\frac{3\sqrt{3}}{2} R^2}$$

(iv) Maximum carried traffic $= \text{total no. of cells } (N_c) * \text{traffic intensity per cell } (A)$ (or of each cell)

$= 411 * 45$

$= 18{,}495$ Erlangs.

1) Total no. of users served for 2% of GOS

$$= \frac{\text{total traffic}}{\text{traffic per user}}$$

The no. of mobiles per unique channel (where it is understood that channels are reused)

$$= \frac{\text{No. of users}}{\text{No. of channels}}$$

$$\frac{\quad}{\text{No. of channels}}$$

$$= \frac{616500}{56 * 12} = 9$$

$\simeq$

no. of channels per cell * no. of cell

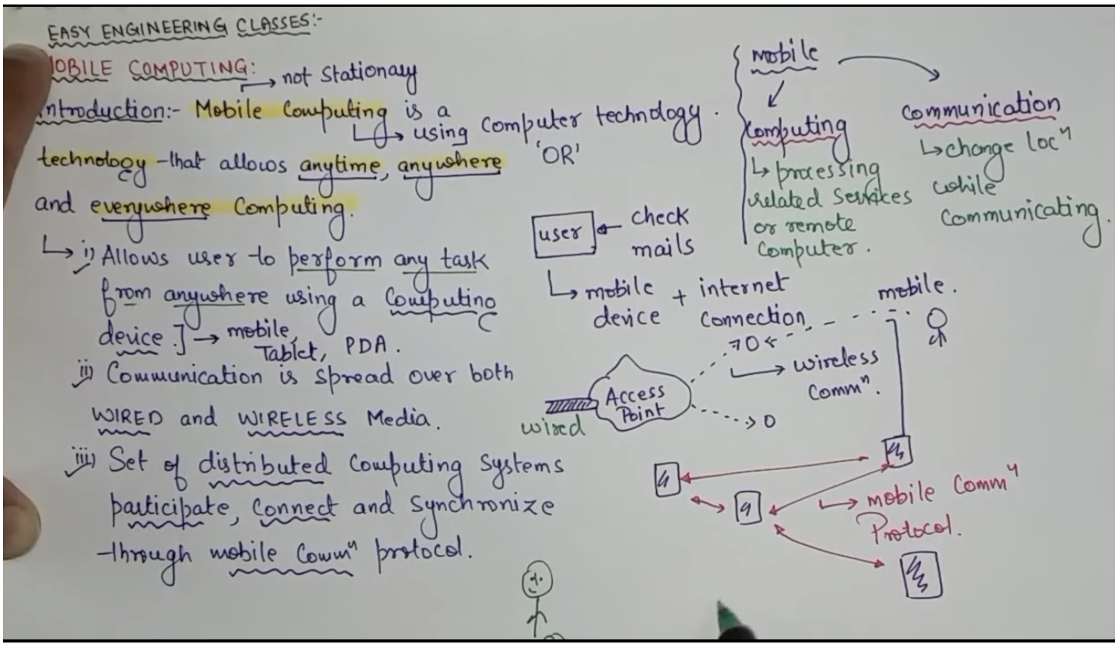
(C) * 12

(extra) Theoretical max^m no. of users that could be served at one time by the system. (or theoretical max^m no. of mobiles served) or total channels in use in system.

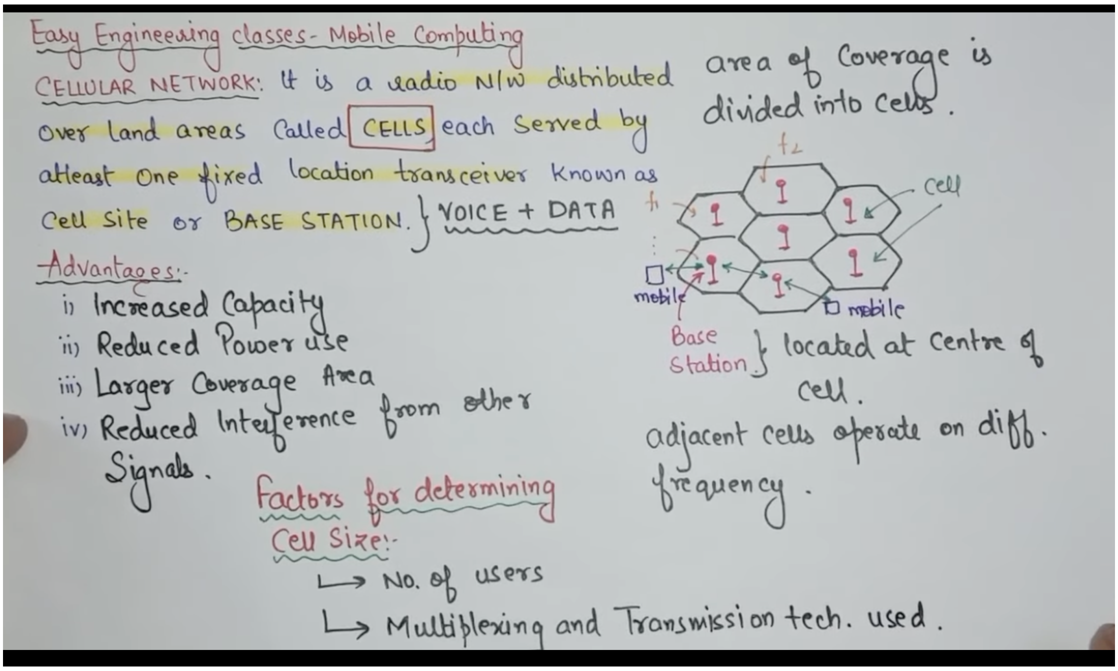
$$= C * N_c = 56 * 411 = 23{,}016.$$

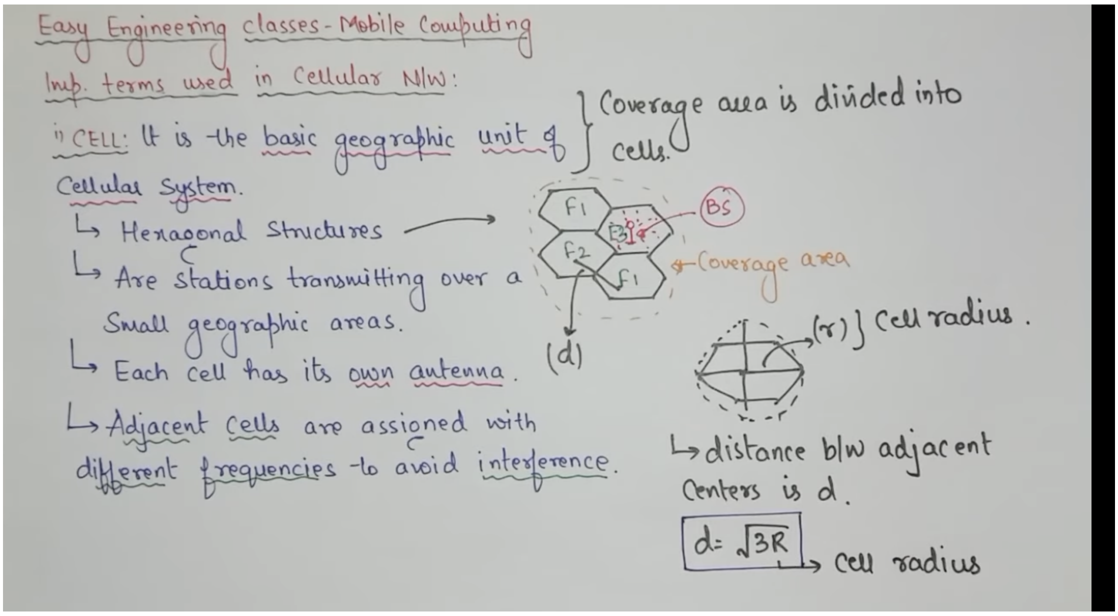
3rd wala question is like 5,11,17 than do * 60 for channels

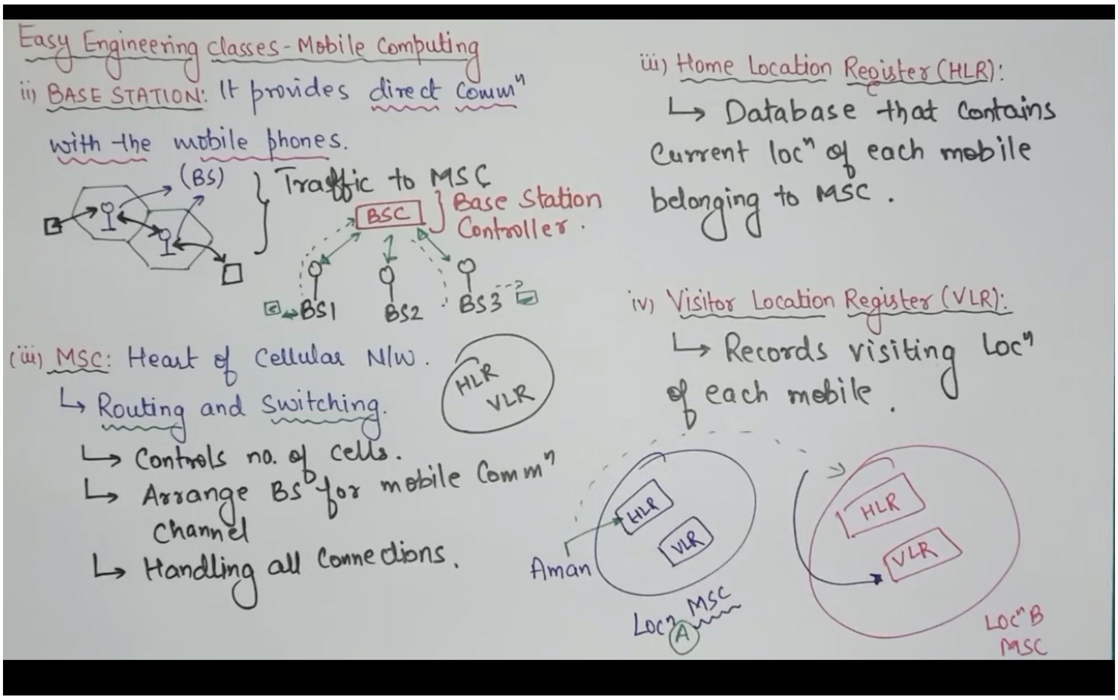
▼ Introduction -

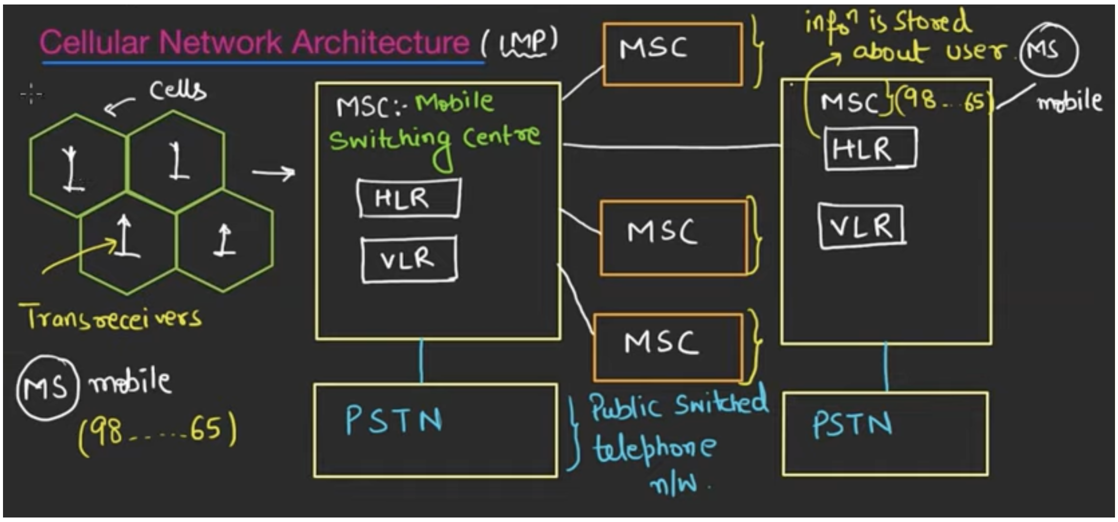


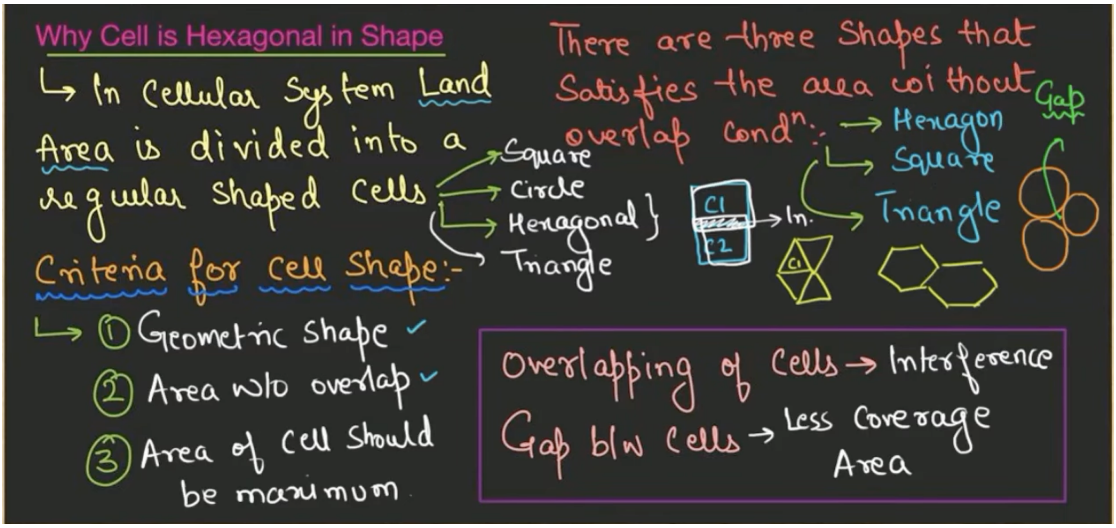
▼ Cellular Network

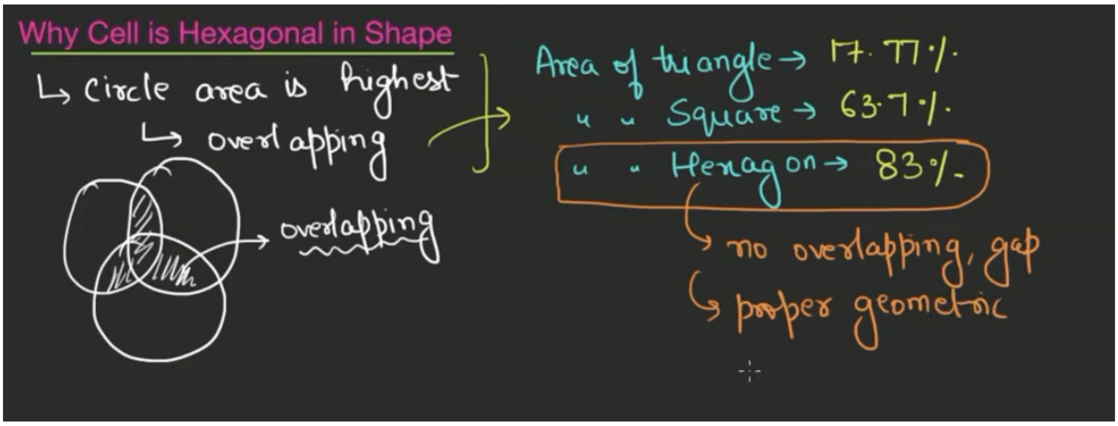


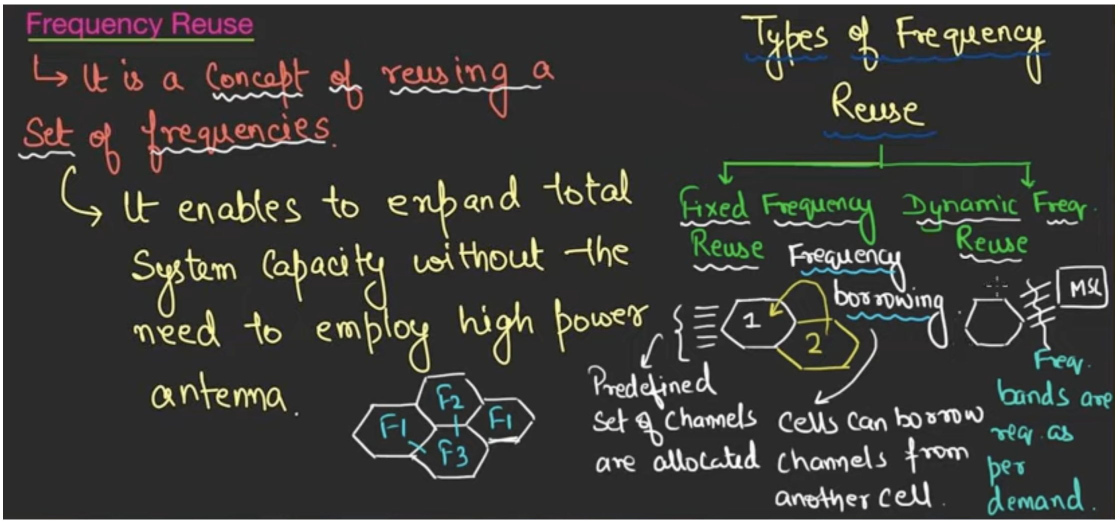

Easy Engineering classes - Mobile Computing
Imp. terms used in Cellular N/W:
i) CELL: It is the basic geographic unit of Cellular System.
Coverage area is divided into cells.
↳ Hexagonal Structures
↳ Are stations transmitting over a small geographic areas.
↳ Each cell has its own antenna.
↳ Adjacent Cells are assigned with different frequencies to avoid interference.
F1
F2
F3
F1
BS
Coverage area
(d)
(r) Cell radius.
↳ distance b/w adjacent centers is d.
d = √3R
cell radius

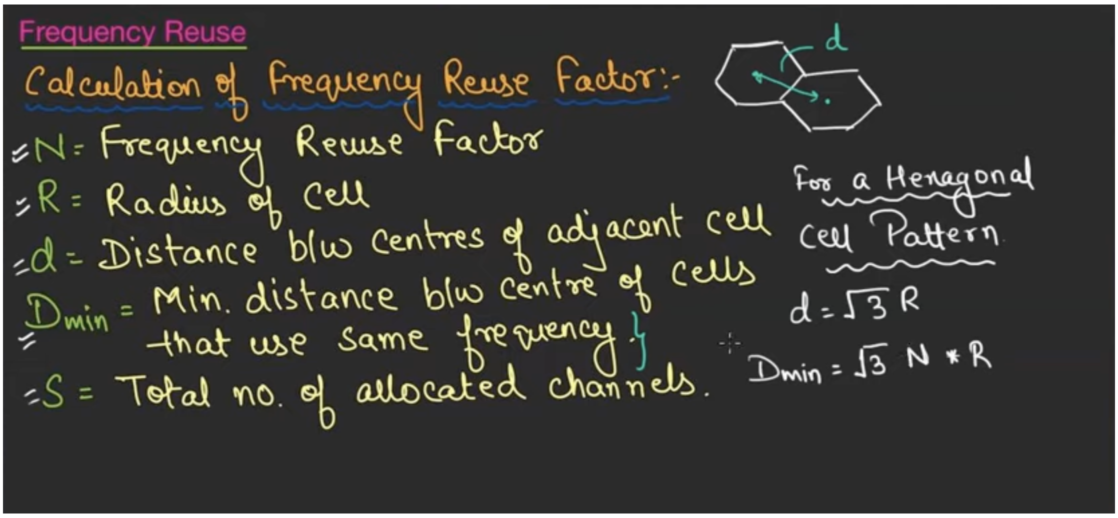

Easy Engineering classes - Mobile Computing
ii) BASE STATION: It provides direct commⁿ with the mobile phones.
(BS)
Traffic to MSC
BSC
Base Station Controller.
BS1
BS2
BS3
(iii) MSC: Heart of Cellular N/W.
↳ Routing and Switching.
↳ Controls no. of cells.
↳ Arrange BS for mobile commⁿ channel
↳ Handling all connections.
HLR
VLR
iii) Home Location Register (HLR):
↳ Database that contains current locⁿ of each mobile belonging to MSC.
iv) Visitor Location Register (VLR):
↳ Records visiting locⁿ of each mobile.
Aman
HLR
VLR
Locⁿ A MSC
HLR
VLR
Locⁿ B MSC


Cellular Network Architecture (IMP)
Cells
Transreceivers
MS mobile
(98.....65)
MSC:- Mobile Switching Centre
HLR
VLR
PSTN
MSC
MSC
MSC
Public Switched telephone n/w.
info^n is Stored about user
MS
MSC (98...65)
mobile
HLR
VLR
PSTN


Why Cell is Hexagonal in Shape
↳ In Cellular System Land Area is divided into a regular shaped cells
Square
Circle
Hexagonal
Triangle
Criteria for cell Shape:-
↳ ① Geometric Shape ✓
② Area w/o overlap ✓
③ Area of cell Should be maximum.
There are three Shapes that Satisfies the area without overlap cond^n:-
→ Hexagon
↳ Square
↳ Triangle
Gap
C1
C2
In.
Overlapping of Cells → Interference
Gap b/w Cells → Less Coverage Area


Why Cell is Hexagonal in Shape
↳ Circle area is highest
↳ overlapping
overlapping
Area of triangle → 17.77%.
" " Square → 63.7%.
" " Hexagon → 83%.
↳ no overlapping, gap
↳ proper geometric



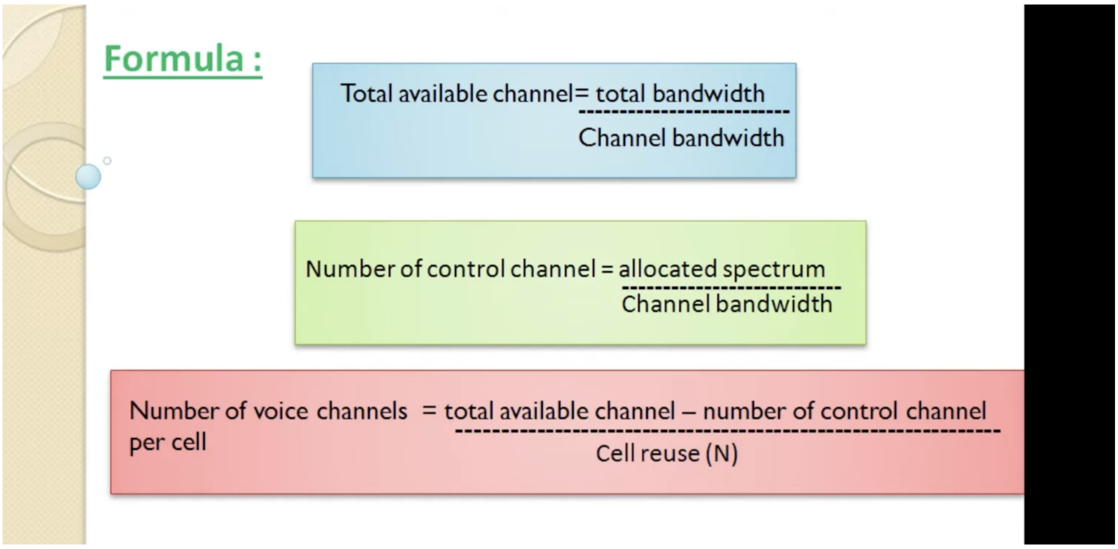


frequency reuse if root (3N) * R and not root (3) N * R

▼ what is channel in cellular technology

In cellular, each cell phone communicates with its base station. Usually, this is done over two channels, one downstream and one upstream. Or sometimes the system uses "time division duplexing" (TDD), which shares one channel for up and down Multiple users in each cell can share the down and up links, with digital protocols such as TDMA (time division), CDMA (code division), or OFDMA and soon FBMC (subcarrier division, I suppose you can call it).
If the two users are on the same cell system, they usually will not be in the same cell. Cells are small. Say, something like 1 or 2 km square, depending. Some can be a lot smaller, especially as we go to 4G and then 5G. Whatever the case may be, cell phones communicate only to base stations, in the normal case.
So, whether cell system users are in the same cellular network or not, most of the time a so-called "backhaul network" becomes involved. The fixed, cabled or wireless network, which ties together the cells and ties the cellular system to the telephone system.

▼ Formulas



▼ Numericals

https://www.youtube.com/watch?v=QlsuBxtmHO8&ab_channel=TECHLECTURE

▼ unit 2

learn gsm -

what is gsm - https://www.youtube.com/watch?v=WAWQCgB53HA&ab_channel=EzEdChannel

gsm basics - https://www.youtube.com/watch?v=PlJFUNcrZ7c&ab_channel=EducationalWordsTouch

auth in gsm - https://www.youtube.com/watch?v=kH2cakCcHCU&ab_channel=TechnicalGuftgu

gsm arch - https://www.youtube.com/watch?v=V-yW7Q6rQmI&ab_channel=TechnicalGuftgu

one more arch - https://www.youtube.com/watch?v=HIQ8Z2xlCS8&ab_channel=AnkitVerma

- gsm ka intro - https://www.youtube.com/watch?v=bfB6opWTLTg&t=16s&ab_channel=PerfectComputerEngineer

- Explain the Authentication and Ciphering procedure of the information exchanged between MS and GSM network.

- Why is the need of Hand off? Explain the procedure of Handover between Cells Controlled by the Same BSC and Handover between Cells Controlled by Different BSCs but the Same MSC/VLR inter- MSC and intra-MSC Handoff.

HANDOVER

The process of changing cells during a call is called handover in GSM terminology. To choose the best target cell, measurements are performed by the MS and the RBS. Because the MS contributes to the handover decision, this type of handover is often called Mobile Assisted HandOver (MAHO).

Another reason for attempting a handover, apart from signal strength and quality, is when the Timing Advance (TA) used by MS exceeds a threshold value set by the operator. This usually happens when the MS is moving over the cell border to another cell.

When the MS has changed cells, the new RBS informs the MS about the new neighboring BCCH carriers so measurements can be taken again. If the MS has also switched to a new LA, a location updating type normal takes place after the call has finished.

Handover can be used for load balancing between cells. During a call setup in a congested cell, the MS can be transferred to a cell with less traffic if an acceptable connection quality is likely to be obtained. Another area where forced handover is a useful tool is maintenance. Channels can be released from traffic if necessary, e.g. for RBS maintenance reasons.

There are several types of handover, including:

- Intra-cell handover
- Handover between cells controlled by the same BSC
- Handover between cells controlled by different BSCs, but the same MSC/VLR
- Handover between cells controlled by different MSC/VLRs

Handover between Cells Controlled by the Same BSC

When performing a handover between two cells controlled by the same BSC, the MSC/VLR is not involved. However, the MSC/VLR will be informed when a handover has taken place. If the handover involves different LAs, location updating is performed once the call is finished.

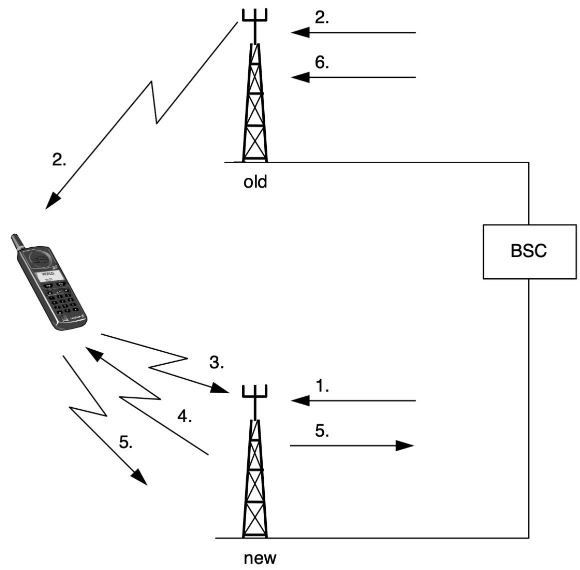


Figure 9-20 Handover: cells controlled by the same BSC

Figure 9-20 Handover: cells controlled by the same BSC

1. The BSC orders the new RBS to activate a TCH.
2. The BSC sends a message to the MS, via the old RBS, containing information about the frequency and time slot to change to and also the output power to use. This information is sent to the MS using FACCH.
3. The MS tunes to the new frequency, and transmits handover access bursts in the correct time slot. Since the MS has no information yet on TA, the handover bursts are very short (only 8 bits of information).
4. When the new RBS detects the handover bursts, it sends information about TA. This is also sent via FACCH.
5. The MS sends a Handover Complete message to the BSC via the new RBS.
6. The BSC tells the old RBS to release the old TCH.

Handover between Cells Controlled by Different BSCs but the Same MSC/VLR

When another BSC is involved in a handover, the MSC/VLR must also be involved to establish the connection between the two BSCs.

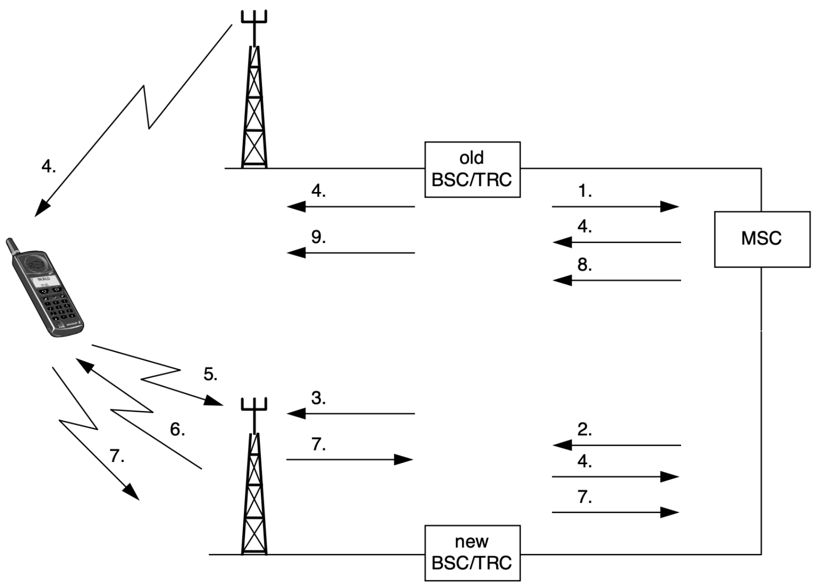


Figure 9-21 Handover: different BSCs but the same MSC/VLR

1. The serving (old) BSC sends a Handover Required message to the MSC containing the identity of the target cell.
2. The MSC knows which BSC controls this cell and sends a Handover Request to this BSC.
3. The new BSC orders the target RBS to activate a TCH.
4. The new BSC sends a message to the MS via the MSC and the old RBS.
5. MS tunes to the new frequency and transmits handover access bursts in the correct time slot.
6. When the new RBS sends information about TA.
7. MS sends a Handover Complete message to MSC via the new BSC.
8. MSC sends the old BSC an order to release the old TCH.
9. The old BSC tells the old RBS to release the TCH.

Handover between Cells Controlled by Different MSC/VLRs

Handover between cells controlled by different MSC/VLRs can only be performed within one PLMN and not between two PLMNs. Cells controlled by different MSC/VLRs also means that they are controlled by different BSCs.

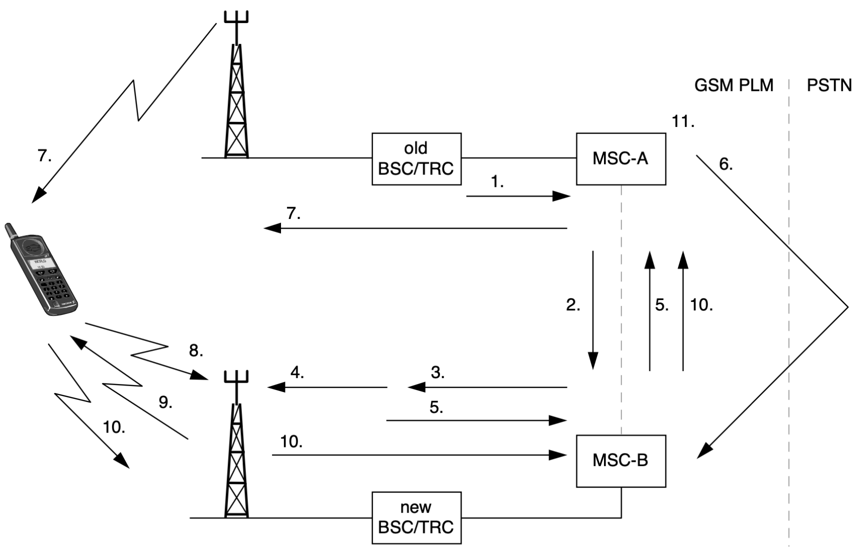


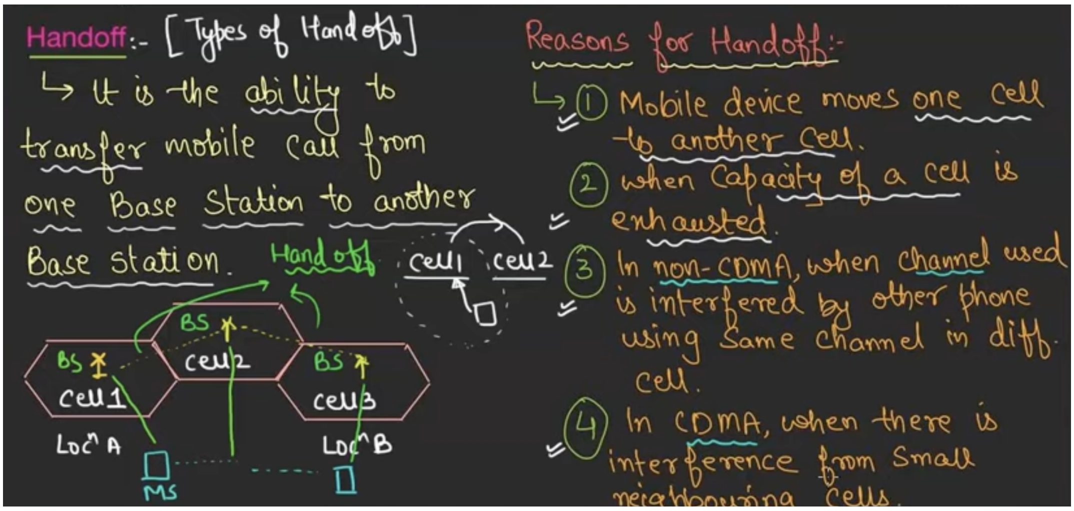
Figure 9-22 Handover: cells controlled by different MSCs

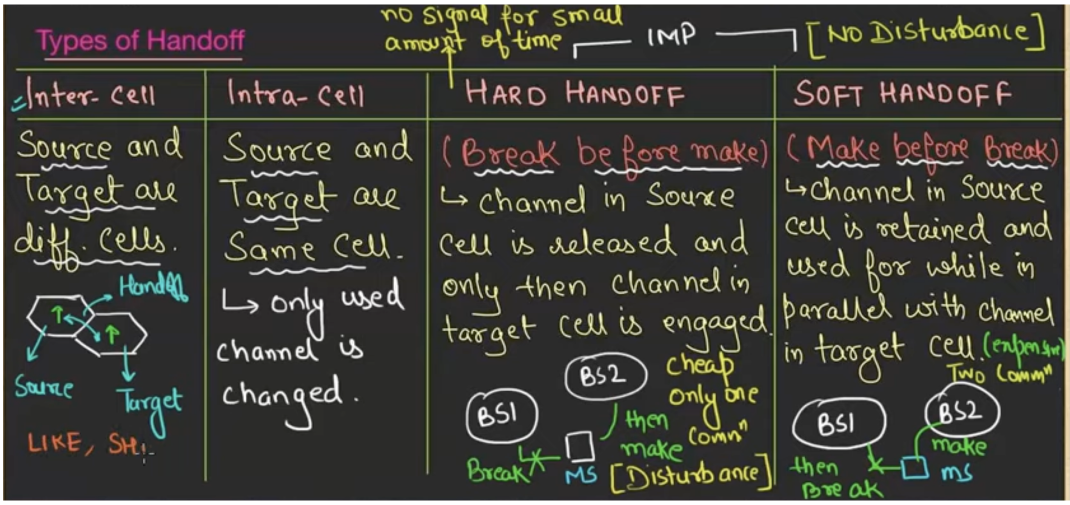
1. The serving (old) BSC sends a Handover Required message to the serving MSC (MSC–A), with the identity of the target cell.
2. MSC–A identifies that this cell belongs to another MSC, (MSC–B), and requests help.
3. MSC–B allocates a handover number to reroute the call. A Handover Request is then sent to the new BSC.
4. The new BSC orders the target RBS to activate a TCH.
5. MSC–B receives the information, and passes it on to MSC–A together with the handover number.
6. A link is set up to MSC–B, possibly via PSTN.
7. MSC–A sends a handover command to the MS, via the old BSC.
8. The MS tunes to the new frequency and transmits handover access bursts in the correct time slot.
9. When the new RBS detects the handover bursts it sends information about TA.
10. The MS sends Handover Complete message to the old MSC via the new BSC and the new MSC/VLR.
11. A new path in the group switch in MSC–A is established, and the call is switched through.
12. The old TCH is deactivated by the old BSC (not shown in the picture).

The old MSC, MSC–A, retains main control of the call until the call is cleared. This is because it contains the information about the subscriber and call details such as charging.

After call release, the MS must perform location updating because an LA never belongs to more than one MSC/VLR service area. The HLR is updated by the VLR–B, and will in turn tell VLR–A to delete all information about the mobile

▼ handoff





- https://www.youtube.com/watch?v=c5NBMwj9ZdA&list=PLBC3G7CyizTrQ22MC96xKTnBX3h2stFbA&index=17&ab_channel=3G4G

- Explain the GSM transmission process in detail.

  it has eight stages -

  - analog to digital conversation which includes sampling, quantization, coding
  - segmentation
  - speech coding
  - channel coding
  - interleaving, then second level
  - ciphering/encryption
  - burst formatting
  - modulation and transmission

STAGE 1: ANALOG TO DIGITAL (A/D) CONVERSION

One of the primary functions of an MS is to convert the analog speech information into digital form for transmission using a digital signal. The analog to digital (A/D) conversion process outputs a collection of bits: binary ones and zeros which represent the speech input.

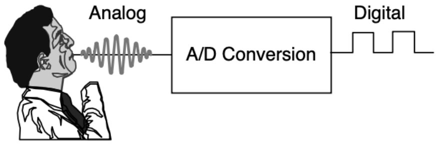


Figure 3-23 A/D conversion

The A/D conversion is performed by using a process called Pulse Code Modulation (PCM). PCM involves three main steps:

- Sampling
- Quantization
- Coding

Step 1: Sampling

Sampling involves measuring the analog signal at specific time intervals.

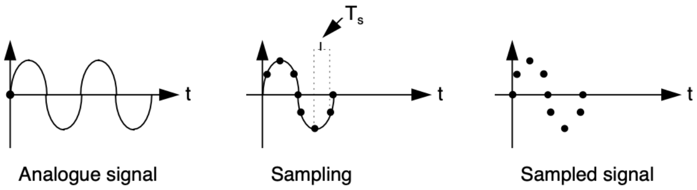


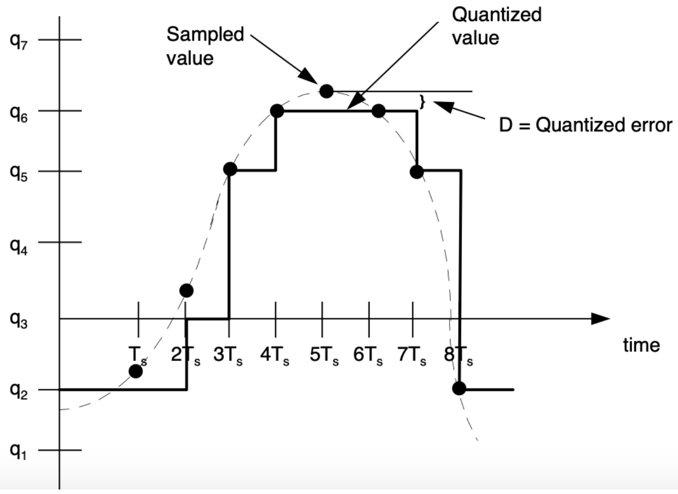
Figure 3-24 Analog signal sampling

To reproduce an analog signal without distortion, the signal must be sampled with at least twice the frequency of the highest frequency component in the analog signal.

Normal speech mainly contains frequency components lower than 3400 Hz. Higher components have low energy and may be omitted without affecting the speech quality much. Applying the sampling theory to analog speech signals, the sampling frequency, should be at least 2 x 3.4 kHz = 6.8 kHz. Telecommunication systems use a sampling frequency of 8 kHz, which is acceptable based on the sampling theory.

Step 2: Quantization

The next step is to give each sample a value. For this reason, the amplitude of the signal at the time of sampling is measured and approximated to one of a finite set of values. The figure below shows the principle of quantization applied to an analog signal. It can be seen that a slight error is introduced in this process when the signal is quantized or approximated. The degree of accuracy depends on the number of quantization levels used. Within common telephony, 256 levels are used while in GSM 8,192 levels are used.



Step 3: Coding

Coding involves converting the quantized values into binary. Every value is represented by a binary code of 13 bits (2^{13} = 8192). For example, a quantized value of 2,157 would have a bit pattern of 0100001101101:

Bit	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Total**
Set to	0	1	0	0	0	0	1	1	0	1	1	0	1	
Value	0	2048	0	0	0	0	64	32	0	8	4	0	1	**2157**

Table 3-2 Coding of quantised value 2157

Summary of A/D Conversion

The result from the process of A/D conversion is 8,000 samples per second of 13 bits each. This is a bit rate of 104 kbits/s.

When it is considered that 8 subscribers use one radio channel, the overall bit rate would be 8 x 104 kbits/s = 832 kbits/s. Recalling the general rule of 1 bit per Hertz, this bit rate would not fit into the 200 kHz available for all 8 subscribers. The bit rate must be reduced somehow - this is achieved using segmentation and speech coding.

STAGE 2: SEGMENTATION AND STAGE 3: SPEECH CODING

☞ Did you know?

In his childhood, Alexander Graham Bell constructed an artificial speaking machine. It was an anatomical model of the human voice tract, complete with teeth, throat, nasal passages and tongue.

By carefully positioning these elements, while simultaneously introducing a sounds source in the throat, Bell was able to articulate simple English words.

The key to reducing the bit rate is to send information about the speech instead of the speech itself. This can be explained with the following analogy:

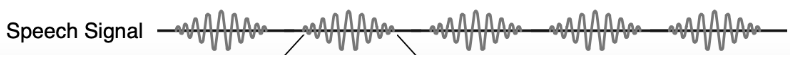
> Person A wishes to listen to a certain piece of music and they know that person B has it on record. A rings B asking for the use of the record for some time. Unfortunately, the record is scratched and cannot be used. Instead, B sends A parameters of how the music is built up - the sheets of music - together with information about how fast it should be played - the frequency - and A reproduces the music.

In GSM, the speech coding process analyzes speech samples and outputs parameters of what the speech consists of: the tone, length of tone, pitch, etc. This is then transmitted through the network to another MS which generates the speech based on these parameters.

The process of segmentation and speech coding is explained in more detail as follows:

The human speech process starts in the vocal chords or speech organs, where a tone is generated. The mouth, tongue, teeth, etc., act as a filter, changing the nature of this tone. The aim of speech coding in GSM is to send only information about the original tone itself and about the filter.

Segmentation: Given that the speech organs are relatively slow in adapting to changes, the filter parameters representing the speech organs are approximately constant during 20 ms. For this reason, when coding speech in GSM, a block of 20 ms is coded into one set of bits. In effect, it is similar to sampling speech at a rate of 50 times per second instead of the 8,000 used by A/D conversion.



Speech Coding[5]: Instead of using 13 bits per sample as in A/D conversion, GSM speech coding uses 260 bits. This calculates as 50 x 260 = 13 kbits/s. This provides a speech quality which is acceptable for mobile telephony and comparable with wireline PSTN phones.

Many types of speech coders are available. Some offer better speech quality, at the expense of a higher bit rate (waveform coders). Others use lower bit rates, at the expense of lower speech quality (vocoders). The hybrid coder which GSM uses provides good speech quality with a relatively low bit rate, at the expense of speech coder complexity.

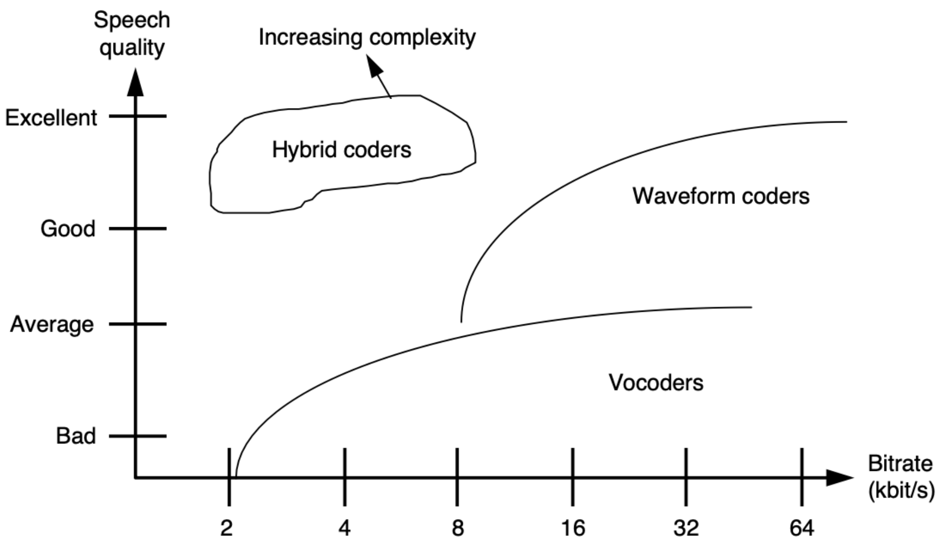


Figure 3-27 Speech quality vs. bit rate

STAGE 4: CHANNEL CODING

Channel coding in GSM uses the 260 bits from speech coding as an input and outputs 456 encoded bits.

The 260 bits are split according to their relative importance:

- Block 1: 50 very important bits
- Block 2: 132 important bits and
- Block 3: 78 not so important bits

The first block of 50 bits is sent through a block coder, which adds three parity bits to result in 53 bits. It is these three bits which are used to detect errors in a received message.

These 53 bits, the 132 bits in the second block and 4 tail bits (total = 189) are sent to a 1:2 convolutional coder which outputs 378 bits. The bits added by the convolutional coder enable the correction of errors when the message is received.

The remaining bits of block 3 are not protected.

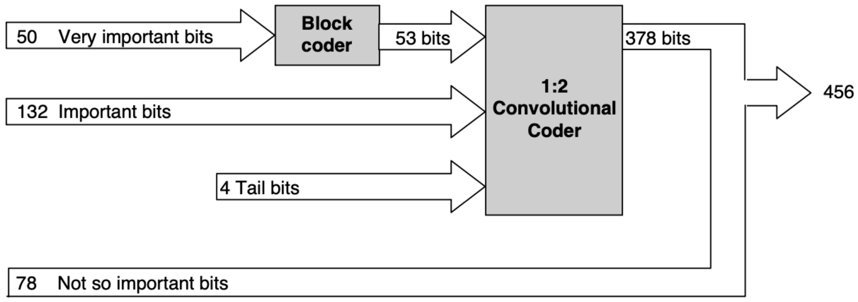


Figure 3-28 Channel coding

STAGE 5: INTERLEAVING

First level of interleaving

The channel coder provides 456 bits for every 20 ms of speech. These are interleaved, forming eight blocks of 57 bits each, as shown in the figure below.

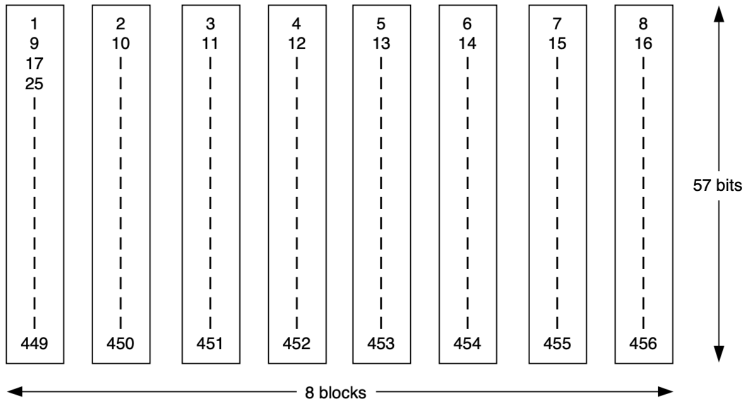


Figure 3-29 Interleaving of 20 ms of encoded speech

As can be seen in Figure 3-30, in any one burst, there is space for two of these blocks. (The remaining bits are explained later in this book.) Thus, if one burst transmission is lost, there is a 25% BER for the entire 20 ms of speech (2/8 = 25%).

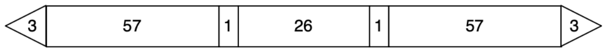


Figure 3-30 Normal burst

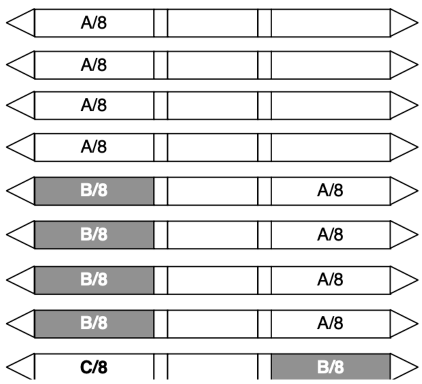
Second level of interleaving

If only one level of interleaving is used, a loss of this burst results in a total loss of 25%. This is too much for the channel decoder to correct. A second level of interleaving can be introduced to further reduce the possible BER to 12.5%.

Instead of sending two blocks of 57 bits from the same 20 ms of speech within one burst, a block from one 20 ms and a block from another 20 ms are sent together. This causes a delay in the system, because the MS must wait for the next 20 ms of speech. However, the system can now afford to loose a whole burst because the loss only affects 12.5% of the bits from each speech frame. This rate can be corrected by a channel decoder.

A	B	C	D
20 ms speech 456 bits = 8x57	20 ms speech 456 bits = 8x57	20 ms speech 456 bits = 8x57	20 ms speech 456 bits = 8x57

Figure 3-31 Speech frame



STAGE 6: CIPHERING/ENCRYPTION

The purpose of ciphering is to encode the burst so that it cannot be interpreted by any device other than the intended receiver. The ciphering algorithm in GSM is called the A5 algorithm. It does not add bits to the burst, meaning that the input and output to the ciphering process is the same as the input: 456 bits per 20 ms.

STAGE 7: BURST FORMATTING

As previously explained, every transmission from an MS/BTS must include some extra information such as the training sequence. The process of burst formatting is to add these bits (along with some others such as tail bits) to the basic speech/data being sent. This increases the overall bit rate, but is necessary to counteract problems encountered on the radio path.

In GSM, the input to burst formatting is the 456 bits received from ciphering. Burst formatting adds a total of 136 bits per block of 20 ms, bringing the overall total to 592.

However, each time slot on a TDMA frame is 0.577 ms long. This provides enough time for 156.25 bits to be transmitted (each bit takes 3.7 μs), but a burst only contains 148 bits. The rest of the space, 8.25 bit times, is empty and is called the Guard Period (GP). This time is used to enable the MS/BTS "ramp up" and "ramp down". To ramp up means to get power from the battery/power supply for transmission. Ramping down is performed after each transmission to ensure that the MS is not using battery power during time slots allocated to other MSs.

The output of burst formatting is a burst of 156.25 bits or 625 bits per 20 ms. When it is considered that there are 8 subscriber per TDMA frame, the overall bit rate for GSM can be calculated to be 270.9 kbits/s.

STAGE 8: MODULATION & TRANSMISSION

The 676 bits per 20 ms of speech must then be sent over the air using a carrier frequency. As previously explained, GSM uses the GMSK modulation technique. The bits are modulated onto a carrier frequency (e.g. 912.2 MHz) and transmitted.

The following figure summarizes the GSM transmission process.

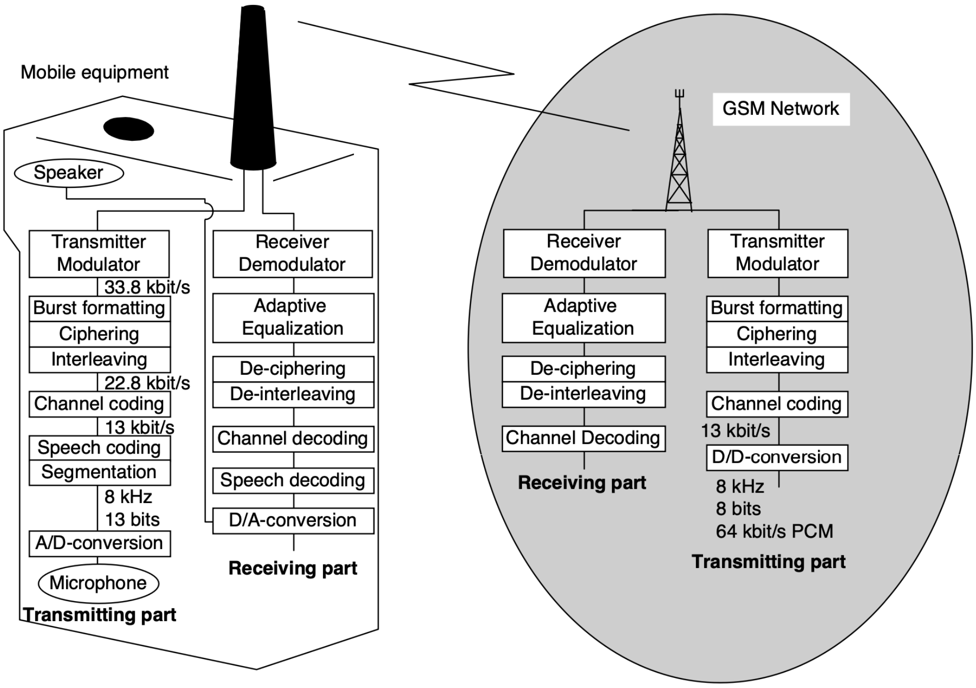


Figure 3-33 GSM transmission process

- Explain in detail about the different logical channels used in GSM network.

INTRODUCTION TO PHYSICAL AND LOGICAL CHANNELS

Each timeslot on a TDMA frame is called a physical channel. Therefore, there are 8 physical channels per carrier frequency in GSM.

Physical channels can be used to transmit speech, data or signaling information.

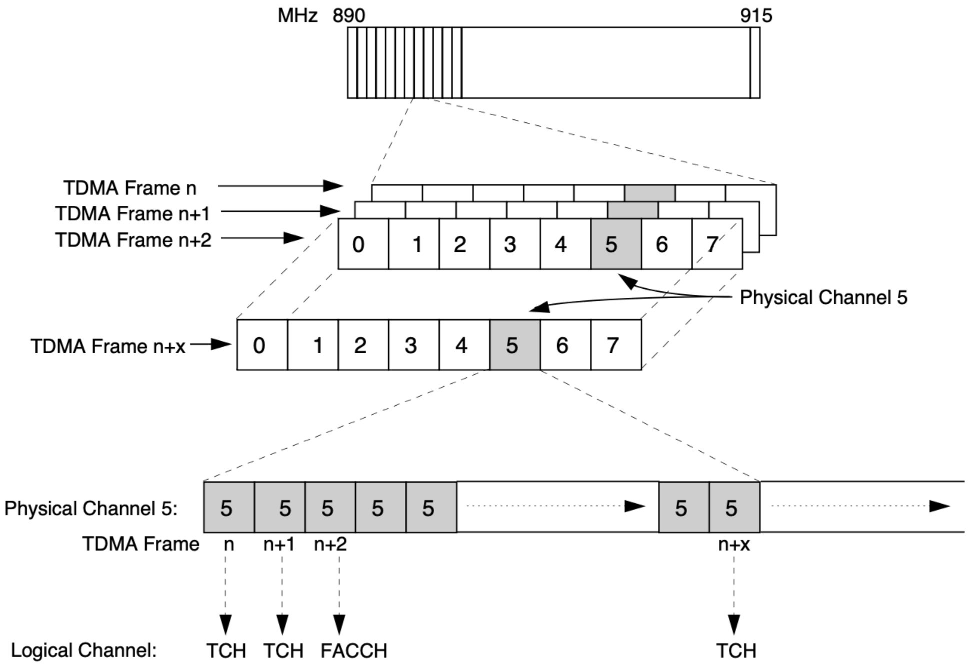


Figure 4-1 The TDMA channel concept

A physical channel may carry different messages, depending on the information which is to be sent. These messages are called logical channels. For example, on one of the physical channels used for traffic, the traffic itself is transmitted using a Traffic CHannel (TCH) message, while a handover instruction is transmitted using a Fast Associated Control Channel (FACCH) message.

LOGICAL CHANNELS

Many types of logical channels exist (see Figure 4-2), each designed to carry a different message to or from an MS.

All information to and from an MS must be formatted correctly, so that the receiving device can understand the meaning of different bits in the message. For example, as seen previously, in the burst used to carry traffic, some bits represent the speech or data itself, while others are used as a training sequence.

There are several types of burst. The relationship between bursts and logical channels is shown in the figure below.

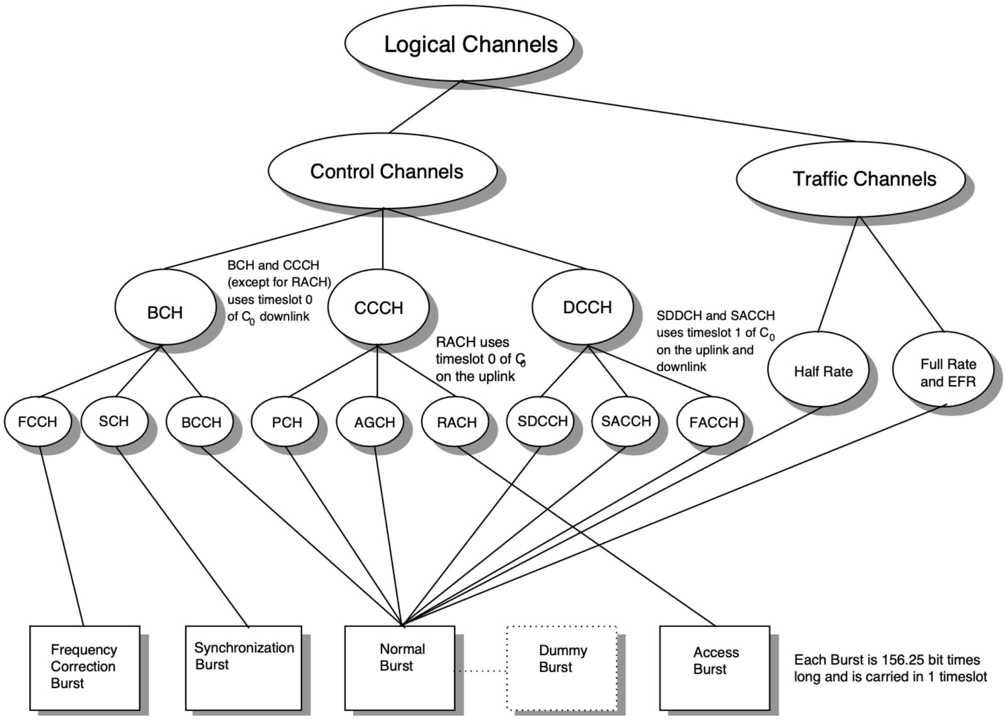


Figure 4-2 Logical channels and bursts

CONTROL CHANNELS

When an MS is switched on, it searches for a BTS to connect to. The MS scans the entire frequency band, or, optionally, uses a list containing the allocated carrier frequencies for this operator. When the MS finds the strongest carrier, it must then determine if it is a control channel. It does so by searching for a particular logical channel called Broadcast Control CHannel (BCCH).

A frequency carrying BCCH contains important information for an MS, including e.g. the current LA identity, synchronization information and network identity. Without such information, an MS cannot work with a network. This information is broadcast at regular intervals, leading to the term Broadcast CHannel (BCH) information.

Broadcast CHannels (BCHs)			
Logical Channel	*Direction*	*BTS*	*MS*
Frequency Correction CHannel (FCCH)	Downlink, point to multipoint	Transmits a carrier frequency.	Identifies BCCH carrier by the carrier frequency and synchronizes with the frequency.
Synchronization CHannel (SCH)	Downlink, point to multipoint	Transmits information about the TDMA frame structure in a cell (e.g. frame number) and the BTS identity (Base Station Identity Code (BSIC)).	Synchronizes with the frame structure within a particular cell, and ensures that the chosen BTS is a GSM BTS - BSIC can only be decoded by an MS if the BTS belongs to a GSM network.
Broadcast Control CHannel (BCCH)	Downlink, point to multipoint	Broadcasts some general cell information such as Location Area Identity (LAI), maximum output power allowed in the cell and the identity of BCCH carriers for neighboring cells.	Receives LAI and will signal to the network as part of the Location Updating procedure if the LAI is different to the one already stored on its SIM. MS sets its output power level based on the information received on the BCCH. Also, the MS stores a list of BCCH

When the MS has finished analyzing the information on a BCH, it then has all the information required to work with a network. However, if the MS roams to another cell, it must repeat the process of reading FCCH, SCH and BCCH in the new cell.

If the mobile subscriber then wishes to make or receive a call, the Common Control CHannels (CCCH) must be used.

Common Control Channels (CCCH)			
Logical Channel	*Direction*	*BTS*	*MS*
Paging CHannel (PCH)	Downlink, point to point	Transmits a paging message to indicate an incoming call or short message. The paging message contains the identity number of the mobile subscriber that the network wishes to contact.	At certain time intervals the MS listens to the PCH. If it identifies its own mobile subscriber identity number on the PCH, it will respond.
Random Access CHannel (RACH)	Uplink, point to point	Receives request from MS for a signaling channel (to be used for call set-up).	Answers paging message on the RACH by requesting a signaling channel.
Access Grant CHannel (AGCH)	Downlink, point to point	Assigns a signaling channel (SDCCH) to the MS.	Receives signaling channel assignment (SDCCH).

Table 4-2 Common Control Channels

At this stage the MS and BSS are ready to begin call set-up procedures. For this the MS and BSS use Dedicated Control CHannels (DCCHs).

Dedicated Control Channels (DCCH)			
Logical Channel	*Direction*	*BTS*	*MS*
Stand alone Dedicated Control CHannel (SDCCH)	Uplink and downlink, point to point	The BTS switches to the assigned SDCCH. The call set-up procedure is performed in idle mode. The BSC assigns a TCH. (SDCCH is also used to transmit text messages).	The MS switches to the assigned SDCCH.Call set-up is performed. The MS receives a TCH assignment information (carrier and time slot).
Cell Broadcast CHannel (CBCH)	Downlink, point to multipoint	Uses this logical channel to transmit short message service cell broadcast.	MS receives cell broadcast messages.
Slow Associated Control CHannel (SACCH)	Uplink and downlink, point to point	Instructs the MS the transmitting power to use and gives instructions on timing advance.	Sends averaged measurements on its own BTS (signal strength and quality) and neighboring BTSs (signal strength). The MS continues to use SACCH for this purpose during a call.
Fast Associated Control CHannel (FACCH)	Uplink and downlink, point to point	Transmits handover information.	Transmits handover request.

Table 4-3 Dedicated Control Channels

Table 4-3 Dedicated Control Channels

TRAFFIC CHANNELS

☞ Did you know?

Enhanced Full Rate (EFR) speech coders improve the speech quality offered across one full rate TCH, but still use a full rate TCH logical channel.

Once call set-up procedures have been completed on the control physical channel, the MS tunes to a traffic physical channel. It uses the Traffic CHannel (TCH) logical channel. There are two types of TCH:

- Full rate (TCH): transmits full rate speech (13 kbits/s). A full rate TCH occupies one physical channel.
- Half rate (TCH/2): transmits half rate speech (6.5 kbits/s). Two half rate TCHs can share one physical channel, thus doubling the capacity of a cell.

- Explain the procedure of the following in GSM network:
  (i) Channel coding

CHANNEL CODING

In digital transmission, the quality of the transmitted signal is often expressed in terms of *how many of the received bits are incorrect*. This is called Bit Error Rate (BER). BER defines the percentage of the total number of received bits which are incorrectly detected.

Transmitted bits	1	1	0	1	0	0	0	1	1	0	
Received bits	1	0	0	1	0	0	1	0	1	0	
Errors		↑					↑	↑			3/10 = 30% BER

This percentage should be as low as possible. It is not possible to reduce the percentage to zero because the transmission path is constantly changing. This means that there must be an allowance for a certain amount of errors and at the same time an ability to restore the information, or at least detect errors so the incorrect information bits are not interpreted as correct. This is especially important during transmission of data, as opposed to speech, for which a higher BER is acceptable.

Channel coding is used to detect and correct errors in a received bit stream. It adds bits to a message. These bits enable a channel decoder to determine whether the message has faulty bits, and to potentially correct the faulty bits.

INTERLEAVING

> ☞ Did you know?
>
> *Interleaving could be compared to sending a group of important people from A to B on different planes. By doing so, the likelihood of losing the entire group is minimised.*

In reality, bit errors often occur in sequence, as caused by long fading dips affecting several consecutive bits. Channel coding is most effective in detecting and correcting single errors and short error sequences. It is not suitable for handling longer sequences of bit errors.

For this reason, a process called interleaving is used to separate consecutive bits of a message so that these are transmitted in a non-consecutive way.

For example, a message block may consist of four bits (1234). If four message blocks must be transmitted, and one is lost in transmission, without interleaving there is a 25% BER overall, but a 100% BER for that lost message block. It is not possible to recover from this.

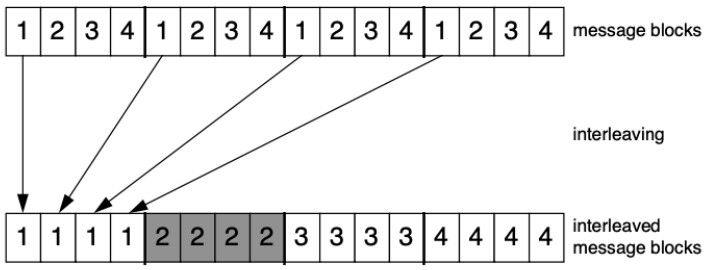


Figure 3-17 Interleaving

If interleaving is used, as shown in Figure 3-18, the bits of each block may be sent in a non-consecutive manner. If one block is lost in transmission, again there is a 25% BER overall. However, this time the 25% is spread over the entire set of message blocks, giving a 25% BER for each. This is more manageable and there is a greater possibility that the errors can be corrected by a channel decoder.

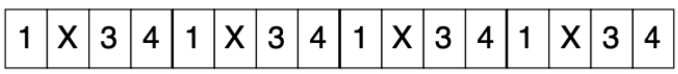


Figure 3-18 Received interleaved message blocks

- Explain about GSM System architecture in detail.
- https://www.cse.iitb.ac.in/~satyajit_01/seminar/node14.html#:~:text=In GSM system the mobile,Base Station Subsystem (BSS).

- Channel Coding - https://www.youtube.com/watch?v=vUtGcsb31zw&ab_channel=ElectronicsSubjectified

▼ unit 3

learn cdma -

cdma basics - https://www.youtube.com/watch?v=EfuK92TEwQY&t=731s&ab_channel=EasyEngineeringClasses

cdma ke baare main - https://www.youtube.com/watch?v=n9-VQvIFvFc&ab_channel=5MinutesEngineering

how signal spread watch code - https://www.youtube.com/watch?v=XJ81CuujwYE&t=560s&ab_channel=CarlOliver

walsh code - https://www.youtube.com/watch?v=BTepizhu4Po&ab_channel=ProfessorRekt

wcdma architecture -

https://www.youtube.com/watch?v=EKlVjJXmBKI&ab_channel=TechnicalGuftgu

https://www.youtube.com/watch?v=syf2EU1q92w&ab_channel=5MinutesEngineering

call registration wcdma -

registration - https://www.youtube.com/watch?v=SFb3UjhGcE4&list=PLBC3G7CyizTrQ22MC96xKTnBX3h2stFbA&index=15&ab_channel=3G4G

attach - https://www.youtube.com/watch?v=SFb3UjhGcE4&ab_channel=3G4G

- What is chipset sequence used in CDMA? Explain the procedure of generation of chipset sequence using:
  (i) Walsh code.
  (ii) Gold Code

- Explain the procedure of End to end call flow in WCDMA.

- Draw the basic architecture of CDMA and explain the function of each components in detail.

- Briefly explain about Internet access to end-user in the cdma2000 network.

WCDMA architecture -

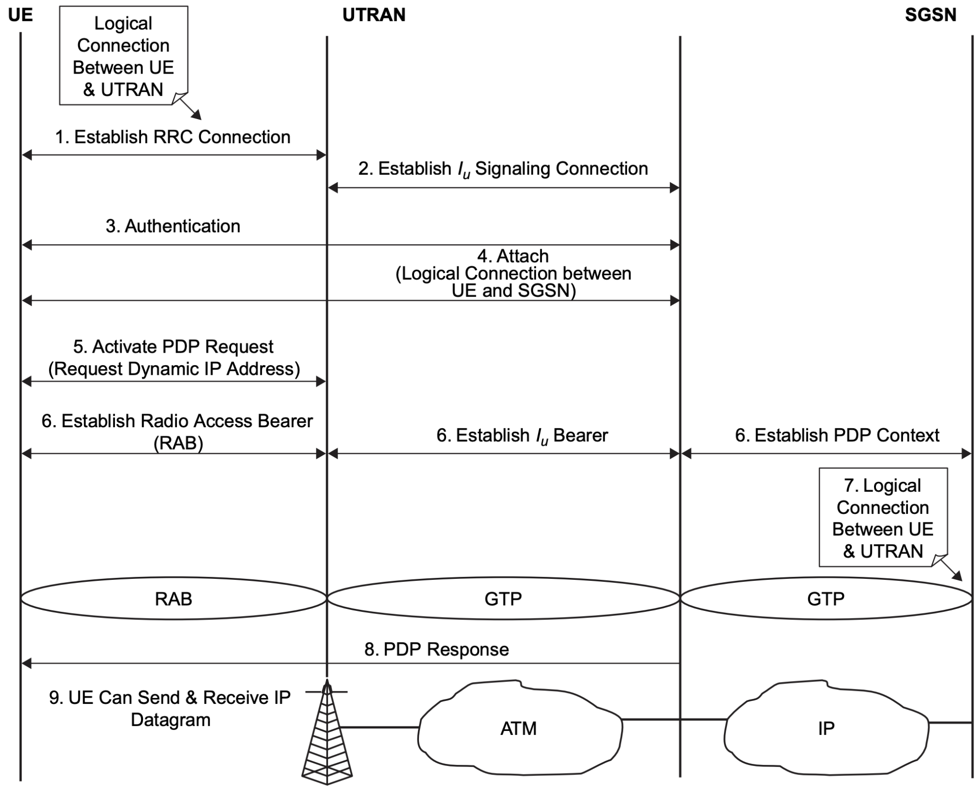
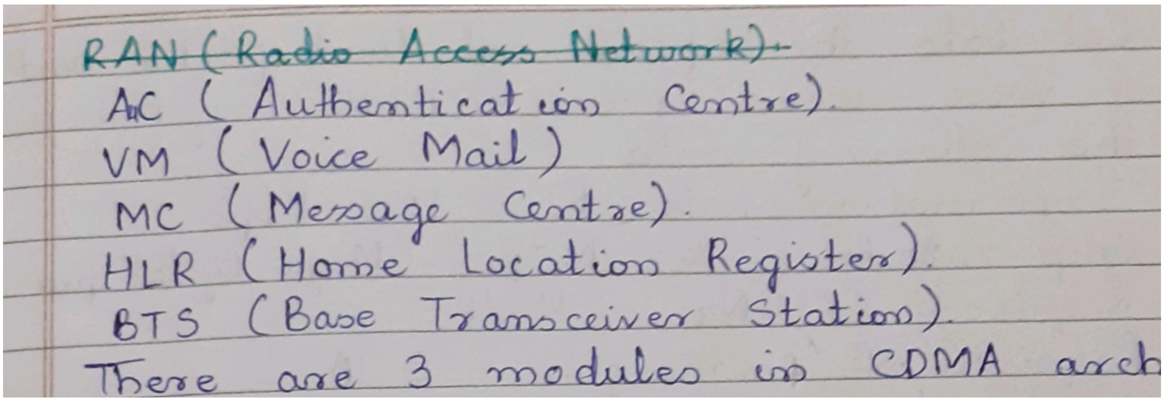


Figure 16.29 End-to-end call flow in WCDMA.

CDMA Architecture -



~~RAN (Radio Access Network).~~
AC (Authentication Centre).
VM (Voice Mail)
MC (Message Centre).
HLR (Home Location Register).
BTS (Base Transceiver Station).
There are 3 modules in CDMA arch

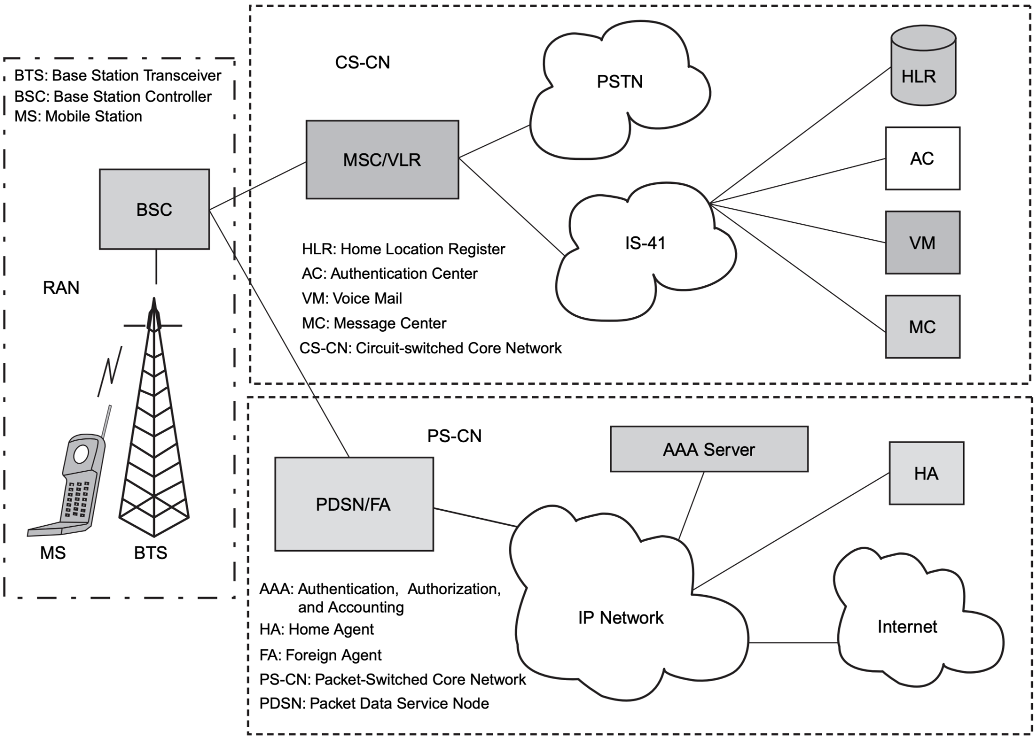
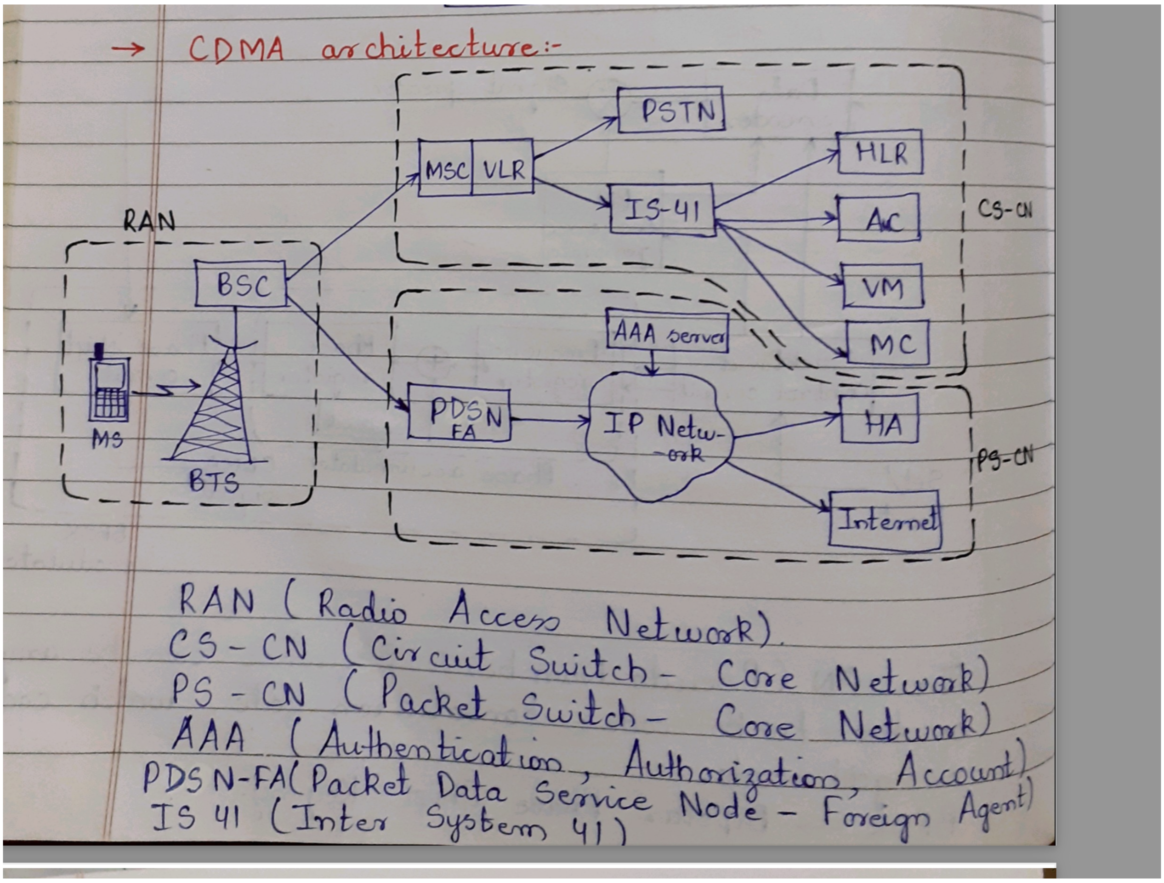
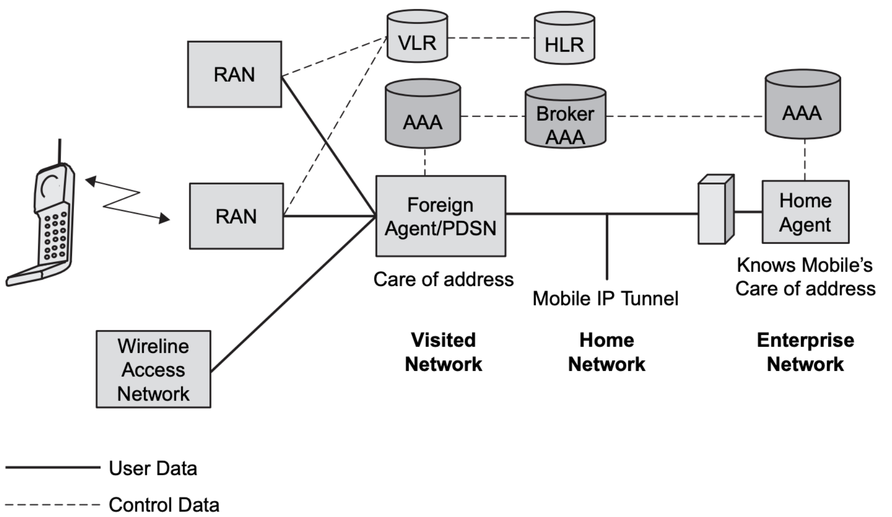


Figure 16.20 cdma2000 network architecture.



- Internet access end user CDMA -



- FA — a router in the visited system to notify HA that the end-user is attached to it. FA delivers datagrams to the end-user.
- HA — a router in the home system to maintain the user current location and to tunnel and deliver user's datagrams when the user is away from home.
- AAA protocol developed in IETF is DIAMETER.

Numericals -

Example 17.7

Calculate the downlink cell load-factor and number of voice users per cell for a WCDMA system using the following data.

- Information rate (R_i): 12.2 kbps
- Chip rate (R_c): 3.84 Mcps
- Required E_b/N_t: 4 dB
- Average interference factor due to other cells: 0.5
- Orthogonality factor (ξ): 0.6
- Interference margin: 3 dB

Solution

$$\eta_{DL} = (\xi + i_{DL}) \cdot \sum_{i=1}^{I} \frac{1}{\frac{R_c}{R_i} \cdot \frac{1}{(E_b/N_t)_{\text{reqd}}} \cdot \frac{1}{\upsilon_i}}$$

$$\text{Load-factor per voice user} = (0.6 + 0.5) \cdot \frac{1}{\frac{3840}{12.2} \cdot \frac{1}{2.512} \cdot \frac{1}{0.65}} = 0.0057$$

$$\text{Interference margin} = 3\,\text{dB} = 2 = \frac{1}{1-\rho};\ \rho = 0.5$$

$$\text{Number of voice users} = \frac{0.5}{0.0057} \approx 87 \text{ per cell}$$

Example 8.1

The coverage gain in dB for the AMR is given as:

$$10\log\left[\frac{\text{DPDCH (kbps)} + \text{DPCCH}}{\text{DPDCH[(AMR bit rate (kbps))]} + \text{DPCCH}}\right]\text{dB}$$

Assuming the power difference between the dedicated physical control channel (DPCCH) and dedicated physical data channel (DPDCH) of the WCDMA to be −3.0 dB for 12.2 kbps AMR speech, calculate the gain in the link budget in dB by reducing the AMR bit rate from 12.2 to 7.95 kbps, and by reducing the AMR bit rate from 12.2 to 4.75 kbps.

Solution

(a)

$$\text{Coverage gain} = 10\log\left[\frac{12.2 + 12.2 \times 10^{-3/10}}{7.95 + 12.2 \times 10^{-3/10}}\right] = 10\log\left[\frac{18.315}{14.064}\right] = 1.15\,\text{dB}$$

(b)

$$\text{Coverage gain} = 10\log\left[\frac{12.2 + 12.2 \times 10^{-3/10}}{4.75 + 12.2 \times 10^{-3/10}}\right] = 10\log\left[\frac{18.315}{10.865}\right] = 2.27\,\text{dB}$$

Example 6.6

Calculate the capacity and spectral efficiency of the DS-CDMA system with an omnidirectional cell using the following data:

- bandwidth efficiency $\eta_b = 0.9$
- frequency reuse efficiency $\eta_f = 0.45$

- capacity degradation factor $c_d = 0.8$
- voice activity factor $v_f = 0.4$
- information bit rate $R = 16.2\,\text{kbps}$
- $E_b/I_0 = 7\,\text{dB}$
- one-way system bandwidth $B_w = 12.5\,\text{MHz}$

Neglect other sources of interference.

Solution

$E_b/I_0 = 5.02$ (7 dB)

$$N_u = \frac{0.45 \times 0.9 \times 0.8 \times 1}{0.4} \times \frac{12.5 \times 10^6}{16.2 \times 10^3 \times 5.02}$$

$N_u = 124.5$ (say 125)

The spectral efficiency, $\eta = \frac{125 \times 16.2}{12.5 \times 10^3} = 0.162$ bits/sec/Hz

In these calculations, an omnidirectional antenna is assumed. If a three sector antenna (i.e., $G = 3$) is used at a cell site with $\lambda = 2.6$, the capacity will be increased to 325 mobile users per cell, and spectral efficiency will be 0.421 bits/sec/Hz.

Example 17.6

Calculate the uplink throughput for data service only for a WCDMA cell using the following information:

- Required E_b/N_t: 1 dB
- Required interference margin: 3 dB (cell loading = 0.5)

- Interference factor due to other cells: 0.5
- Channel activity factor: 1.0

Solution

$$\text{Load factor} = M(1+\beta)\cdot\frac{1}{1+\frac{R_c}{R_i}\cdot\frac{1}{(E_b/N_t)_{\text{reqd}}}\cdot\frac{1}{\upsilon_i}} \approx M(1+\beta)\cdot\frac{1}{\frac{R_c}{R_i}\cdot\frac{1}{(E_b/N_t)_{\text{reqd}}}\cdot\frac{1}{\upsilon_i}}$$

$$\text{Throughput} = R_i \times M = \frac{\text{cell loading} \times R_c}{(E_b/N_t)_{\text{reqd}}\cdot(1+\beta)} = \frac{0.5\times 3840}{1.259\times 1.5} = 1016\,\text{kbps}$$

T 3.9:

Using a bit string of 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 1 (first bit) demonstrate operations of 4 × 4 bit interleaver/de-interleaver in converting a burst error into bit errors.

Solution

Input to the interleaver:

$$0\,0\,0\,0\,0\,1\,1\,1\,0\,0\,0\,1\,0\,0\,0\,1 \rightarrow \begin{bmatrix}1&0&0&0\\1&0&0&0\\1&1&1&0\\0&0&0&0\end{bmatrix} \text{write the data in rows}$$

Output of the interleaver: 0 0 0 0 0 1 0 0 0 1 0 0 0 1 1 1

$$\text{Input to the de-interleaver: } 0\,0\,0\,0\,0\,1\,0\,\underline{0}\,\underline{0}\,\underline{1}\,\underline{0}\,0\,0\,1\,1\,1 \rightarrow \begin{bmatrix}1&1&1&0\\0&\underline{0}&\underline{1}&\underline{0}\\\underline{0}&0&1&0\\0&0&0&0\end{bmatrix}$$

Burst error is indicated by italic bold-face (under-score).

The output of the de-interleaver: 0 0 $\underline{0}$ 0 0 1 $\underline{1}$ 1 0 0 $\underline{0}$ 1 0 $\underline{0}$ 0 1

▼ unit 4

- Major Components of iot -

1.4.2 Major Components of IoT System

Major components of IoT devices are:

1. **Physical object** with embedded software into a hardware.
2. **Hardware** consisting of a microcontroller, firmware, sensors, control unit, actuators and communication module.
3. **Communication module:** Software consisting of device APIs and device interface for communication over the network and communication circuit/port(s), and middleware for creating communication stacks using 6LowPAN, CoAP, LWM2M, IPv4, IPv6 and other protocols.
4. **Software** for actions on messages, information and commands which the devices receive and then output to the actuators, which enable actions such as glowing LEDs, robotic hand movement etc.

- Cisco Reference Model -

Level 7- Collaboration and Processes (Involving people and business processes)

Level 6- Application (Reporting, Analysis, Control)

Level 5- Data Abstraction (Aggregation and Access)

Level 4- Data Accumulation (Storage)

Level 3- Edge Computing (Data Element Analysis and Transformation)

Level 2- Connectivity (Communication and Processing Units)

Level 1- Physical Devices and Controllers (the things in IoT)
[Sensors, machines, devices, intelligent edge nodes of different types]

CISCO seven leveled reference model

- Platform and Integration tools -

1.4.5 Platforms and Integration Tools

ThingSpeak is an open data platform with an open API. It consists of APIs that enable real-time data collection, geolocation data, data processing and visualisations. It enables device status messages and plugins. It can process HTTP requests and store and process data. It can integrate multiple hardware and software platforms. It supports Arduino, Raspberry Pi,[9] ioBridge/RealTime.io, and Electric Imp. An important feature of ThingSpeak is the support to MATLAB data analytics, mobile, web applications and social networks.

Nimbits is a cloud platform which supports multiple programming languages, including Arduino, JavaScript, HTML or the Nimbits.io Java library.[10] The software deploys on Google App Engine, any J2EE server on Amazon EC2 or Raspberry Pi. It processes a specific type of data and can also store the data. The data can be time- or geo-stamped.

IoT Toolkit offers Smart Object API, HTTP-to-CoAP Semantic mapping and a variety of tools for integrating multiple IoT-related sensor networks and protocols.[11]

SiteWhere provisions a complete platform for managing IoT devices. It enables gathering of data and integrating it with external systems. SiteWhere can be used on Amazon's cloud or downloaded. It also integrates MongoDB, ApacheHBase and multiple big data tools.[12]

Other platforms are Microsoft Azure,[13] TCS connected universe platform (TCS CUP),[14] Xively,[15] smartliving[16], thethings.io[17] and exosite.[18] Usage of Xively and Nimbits cloud are described in detail in Chapter 6. Details of TCS CUP are described in Chapter 12.

- Machine2Machine -

1.6 M2M COMMUNICATION

LO 1.6 Outline the functions of M2M architectural domains and relationships of an M2M system with an IoT system

Machine-to-machine (M2M) refers to the process of communication of a physical object or device at machine with others of the same type, mostly for monitoring but also for control purposes. Each machine in an M2M system embeds a smart device. The device senses the data or status of the machine, and performs the computation and communication functions. A device communicates via wired or wireless systems. The communication protocols are 6LowPAN, LWM2M, MQTT, and XMPP. Each communication device is assigned 48-bits Ipv6 address.

1.6.1 M2M to IoT

IoT technology in industry involves the integration of complex physical machinery M2M communication with the networks of sensors, and uses analytics, machine learning, and knowledge discovery software.

M2M technology closely relates to IoT when the smart devices or machines collect data which is transmitted via the Internet to other devices or machines located remotely. The close difference between M2M and IoT is that M2M must deploy device to device, and carry out the coordination, monitoring, controlling of the devices and communicate without the usage of Internet whereas IoT deploys the internet, server, internet protocols and server or cloud end applications, services or processes.

M2M has many applications in fields such as industrial automation, logistics, smart grid, smart cities, health and defence. Initial applications of M2M were found in automation and instrumentation only, but now these include telemetric applications and Industrial Internet of Things (IIoT) as well.

Problem
Give examples of usages of M2M and IIoT.

Solution
1. Examples of M2M usages are coordinated movement of tools, robots, drones, refinery operations and sequential control at each stage during manufacturing. These include manufacturing of food packets, assembly in assembly lines, tracking of failures along the railway tracks etc.
2. IIoT finds applications in the fields of manufacturing at multiple locations, railways, mining, agriculture, oil and gas, utilities, transportation, logistics and healthcare services along with usages of the internet, and usages of software for analytics, machine learning, and knowledge discovery in these areas.

m2m and iot - https://www.youtube.com/watch?v=BqKZKvFgv1U&list=PLBC3G7CyizTrQ22MC96xKTnBX3h2stFbA&index=18&ab_channel=3G4G

- M2M architecture -

M2M architecture consists of three domains (Figure 1.9):
1. M2M device domain
2. M2M network domain
3. M2M application domain

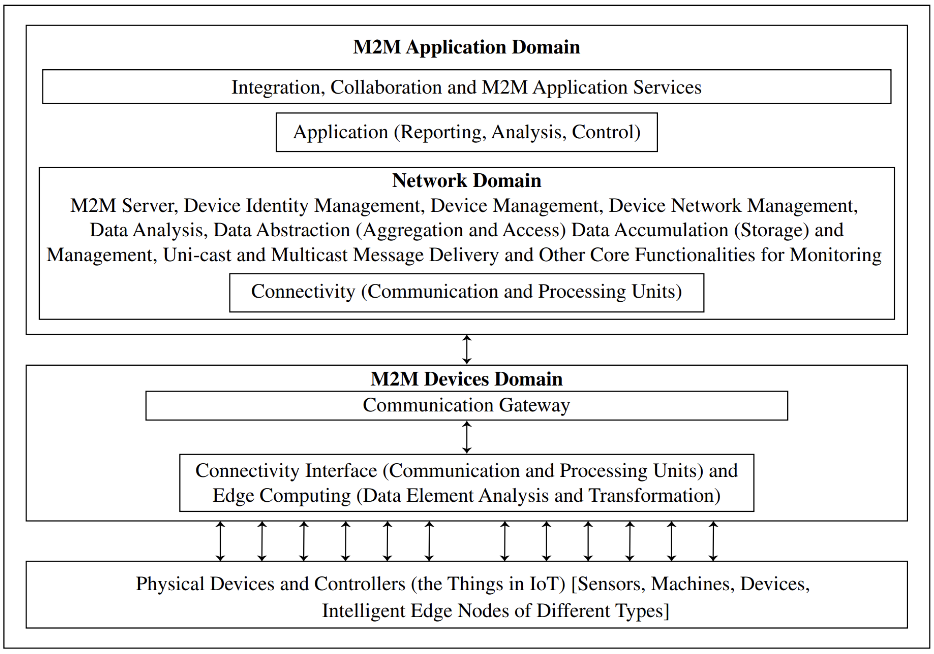


Figure 1.9 Three domains of M2M architecture

- CoAP -

3.2.1 Constrained Application Protocol

IETF recommends Constrained Application Protocol (CoAP) which is for CoRE using ROLL data network. Features of CoAP are:

- An IETF defined application-support layer protocol
- CoAP web-objects communicate using request/response interaction model.
- A specialised web-transfer protocol which is used for CoRE using ROLL network.
- It uses object-model for the resources and each object can have single or multiple instances.
- Each resource can have single or multiple instances.
- An object or resource use CoAP, DTLS (security binding with PSK, RPK and Certificate) and UDP protocols for sending a request or response.
- Supports the resource directory and resource-discovery functions
- The resource identifiers use the URIs as follow coap://… .
- Has small message header, 4 bytes are for ver (version), T (message type) [Types are confirmable, non-confirmable, acknowledgement and reset], TKL (token length), code (request method or response code), message ID 16-bit identifier, token (optional response matching token). Confirmable means server must send an acknowledgement, while non-confirmable means no acknowledgement is required.
- CoRE communication is asynchronous communication over the ROLL.
- Integrates easily with the web using CoAP application cross-protocol proxies. This is facilitated because HTTP and CoAP, both share the REST model. A web client may not even be aware that it just accessed an IoT device sensor resource or sent data to an IoT device actuator resource.
- Use of REST: The access by a CoAP object or its resource is thus using:

- Restful HTTP -

3.4.4 REST and RESTful HTTP Web Applications

RESTful APIs

W3C Technical Architecture Group (TAG) developed the Representational State Transfer (REST) architectural style.[11] The group worked in parallel with HTTP 1.1. REST is a coordinated set of constraints which are used during design of software components in a distributed hypermedia, and the design depends on the characteristics of stateless, client-server, cacheable communication using a protocol. World Wide Web uses REST practices and constraints. REST is a simpler alternative for SOAP and Web Services Description Language (WSDL). REST style web resources and Resource-Oriented Architecture (ROA) have increasingly replacing SOAP.

[11] https://www.w3.org/2001/sw/wiki/REST

interaction when the client selects those links and initiates a new state transition.

RESTful

When all interactions used in the applications conform fully to the REST constraints then these are called RESTful. RESTful APIs comply with these constraints and thus conform to the REST architectural style. Web services with RESTful APIs adhere to the REST architectural constraints.

REST architectural style can be used for HTTP access by GET, POST, PUT and DELETE methods for resources and building web services.

RESTful HTTP APIs

Standard HTTP methods are GET, PUT, POST and DELETE. HTTP based RESTful APIs use the following:

- URIs/URLs, such as http://weatherMsgService.com/weatherMsg/ and hypertext links to reference state and reference related resources, and JSON, TLV or an Internet media type (MIME type) hypertext links
- REST-based web objects communicate typically, but not always, over the HTTP. The World Wide Web itself represents the largest implementation of a system conforming to the REST architectural style. RESTful HTTP system feature is that communication is over the HTTP and use verbs (commands) same as in HTTP, namely GET, POST, PUT and DELETE.

- MQTT Protocol -

MQTT Protocol

Message Queuing Telemetry Transport (MQTT) is an open-source protocol for *machine-to-machine (M2M)/IoT connectivity*. Word 'telemetry', in English dictionary, means measuring and sending values or messages to far off places by radio or other mechanism.

IBM first created it and then donated it to M2M 'Paho' project of Eclipse. A version is MQTT v3.1.1. MQTT has been accepted (2014) as OASIS (Organization for the Advancement of Structured Information Standards) standard[6] MQTT protocol is used for connectivity in M2M/IoT communication.

A version is MQTT-SN v1.2. Sensor networks and non-TCP/IP networks, such as ZigBee can use the MQTT-SN. MQTT-SN is also a publish/subscribe messaging protocol. It enables extension of the MQTT protocol for WSNs, the sensor and actuator devices and their networks.

Recall from Figure 1.3, IBM conceptual framework for the IoT, which showed MQTT applications for one of the communication management functions. Figure 3.6 shows messages interchange between M2M/IoT device objects (publisher and subscriber) and web objects (publisher and subscriber) using an MQTT broker.

- XMPP

XML is an open source language used for encoding text and messages

XMPP

XMPP is an XML-based specification for messaging and presence protocols. XMPP is also an open-source protocol recommended specification which is accepted by IETF. RFC is an international organisation and stands for 'Recommended for Comments'. RFC 6120 document specifies the XMPP for CoRE. Another recommendation, RFC 6121 XMPP specifies the instant messaging (IM) and presence, and RFC 6122 XMPP specifies the (message) address format.

Messages notify *presence* for the IMs to one or many at the same time. It enables chatting and Multi-User Chat (MUC) after creation of a chat room, where different users can do the IMs. XMPP enables interoperable communication: for example, Google Talk. XMPP enables IMs between many users as it uses presence-notifications and chat features.

Chat room is an application, in which all those who have subscribed (meaning persons and objects initiating chatting and messaging to one another at the same time) are provided a room-like view and use the IMs among themselves.

XMPP is extensible—XSF (XMPP standards foundation) develops and publishes the xeps (XMPP extension protocols). The xeps enable the addition of features and new applications. List of XMPP xeps for web objects is quite long. Examples of xeps are:

- xep-DataForms Format
- xep-XHTML-IM
- xep-Service Discovery
- xep-MUC
- xep-Publsih-Subscribe and Personal Eventing Protocol
- xep- File Transfer
- xep-Jingle for Voice and video

- Architecture of cloud platform using Iot -

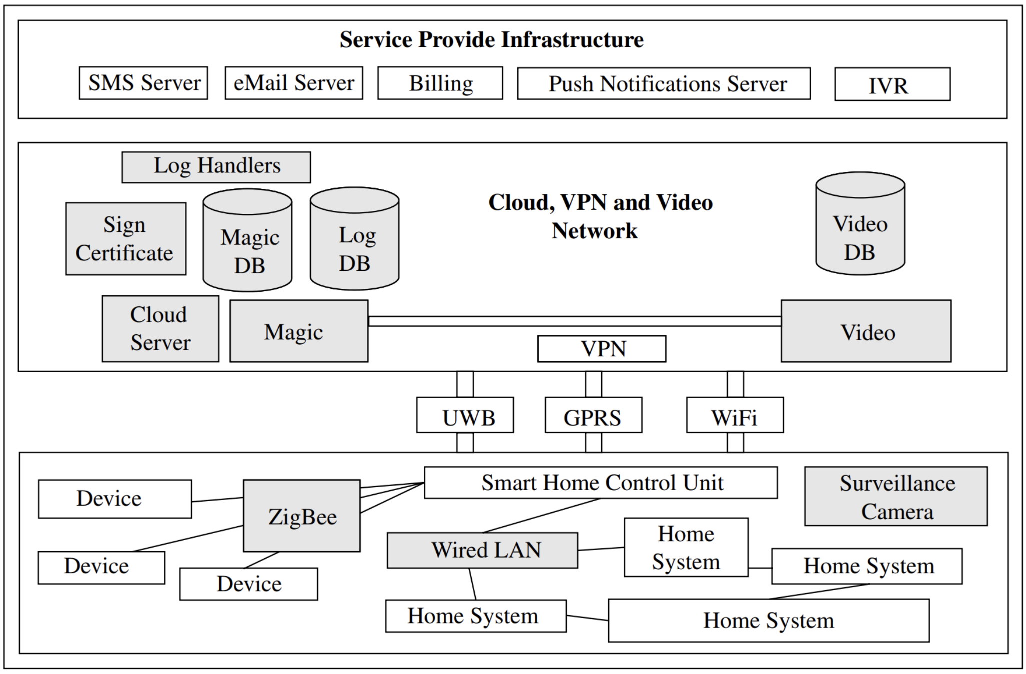


Figure 1.11 An architectural view of cloud (named Magic)-based IoT platform for smart home

- Features of arduino uno board -

Table 8.1 Features of UNO and other IoT device and wearable device boards

Board/ Shield	Application	AVR® Micro-controller/ Clock	Operating/ input V	EEPROM/ SRAM/ Flash	Analog In/ Out/Digital IOs/*n*-bit PWM	USB/ UART	Ethernet/ Wi-Fi/ GSM
Due	Fast computations, ARM based MCU	ATSAMSX8I	3.3 V/ 7 V–12 V	0 kB/ 96 kB/ 512 MB	12/2/54/ 12	2 micro/4	0/0/0
UNO	Getting started with electronics and coding	ATMega328/ 16 MHz	5 V/7 V–12 V	1 kB/2 kB/ 32 kB	6/0/14/6	Standard/ 1	0/0/0
Yun	IoT	(i) ATmega 32U4/ 16MHz (ii) AR9331 Linux/ 400	5 V	(i) 1 kB/ 2.5 MB/ 32 MB (ii) 1 kB/	(i) 20/7/ 12/0	Micro/1	0/0/0

(*Contd.*)

	nics and coding						
Yun	IoT	(i) ATmega 32U4/ 16MHz (ii) AR9331 Linux/ 400 MHz	5 V	(i) 1 kB/ 2.5 MB/ 32 MB (ii) 1 kB/ 16 MB/ 64 MB	(i) 20/7/ 12/0	Micro/1	0/0/0
Ethernet	IoT	ATMega328/ 16 MHz	5 V/ 7 V–12 V	1 kB/2 kB/ 32 kB	6/0/14/4	Standard/ 0	1/0/0
Fio	IoT	ATMega328P/ 8 MHz	3.3 V / 3.7 V–7 V	1 kB/2 kB/ 32 kB	8/0/14/6	Mini/1	0/0/0
Gemma	Wearable	ATtiny85/8 MHz	3.3 V/ 4–16 V	512 B/512 B/8 kB	1/0/3/2	Micro/0	0/0/0
LilyPad	Wearable	ATmega168V/ 8 MHz ATmega328P/ 8 MHz	2.7–5.5 V/ 2.7–5.5 V	512 B/1 kB/ 16 kB	6/0/14/6	0/0	0/0/0
LilyPad SimpleSnap	Wearable	ATmega328P/ 8 MHz	2.7–5.5 V/ 2.5–5.5 V	512 B/ 512 B/8 kB	4/0/9/4	0/0	0/0/0
LilyPadUSB	Wearable	ATmega32U4/ 8 MHz	3.3 V/ 3.8–5 V	1 kB/ 2.5 kB/32 kB	4/0/9/4	Micro/0	0/0/0

Table 8.2 lists the features of ARM based Arduino for IoT devices and wearable devices board, 'due' for fast computations and communication.

- Architectures of sensors and actuators with Arduino boards for Internet of Streetlights applications.-

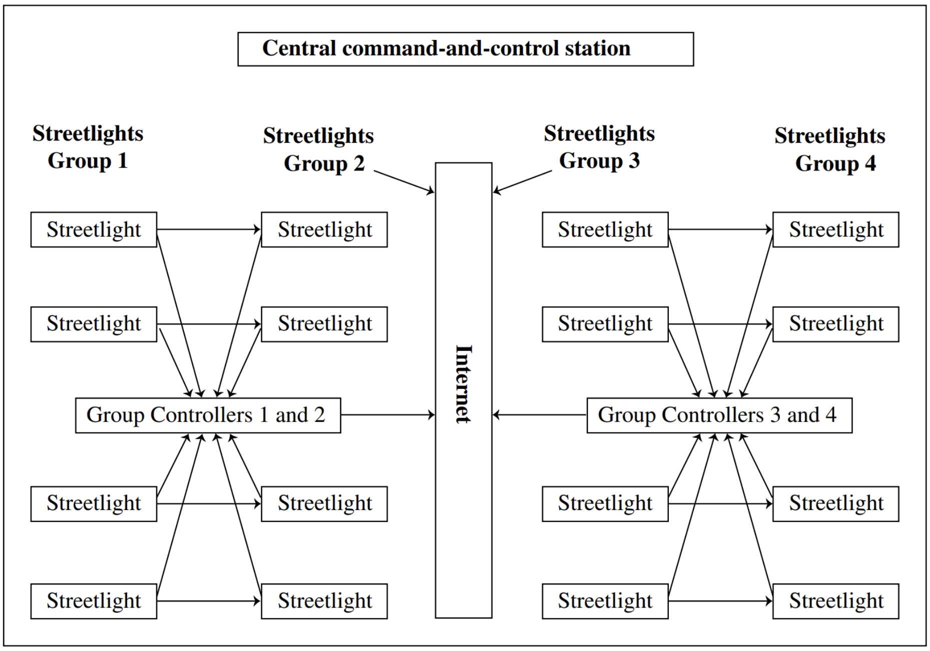


Figure 1.1 Use of Internet of Things concept for streetlights in a city

1.1.3 Smart and Hyperconnected Devices

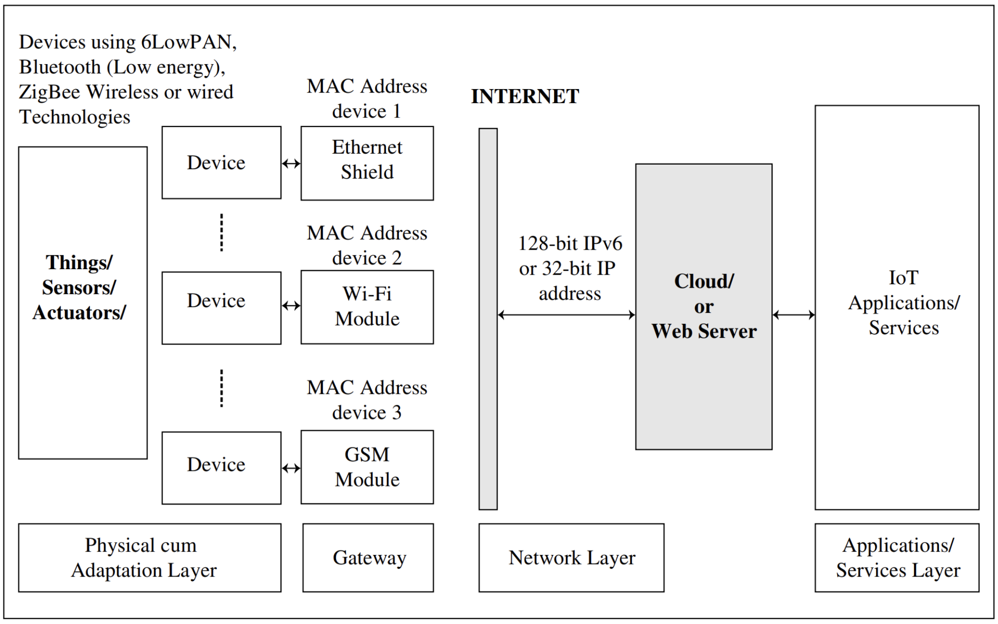


Figure 8.4 'Things', Sensors, actuators, devices and circuit boards connected to Internet and cloud or web-server for applications and services

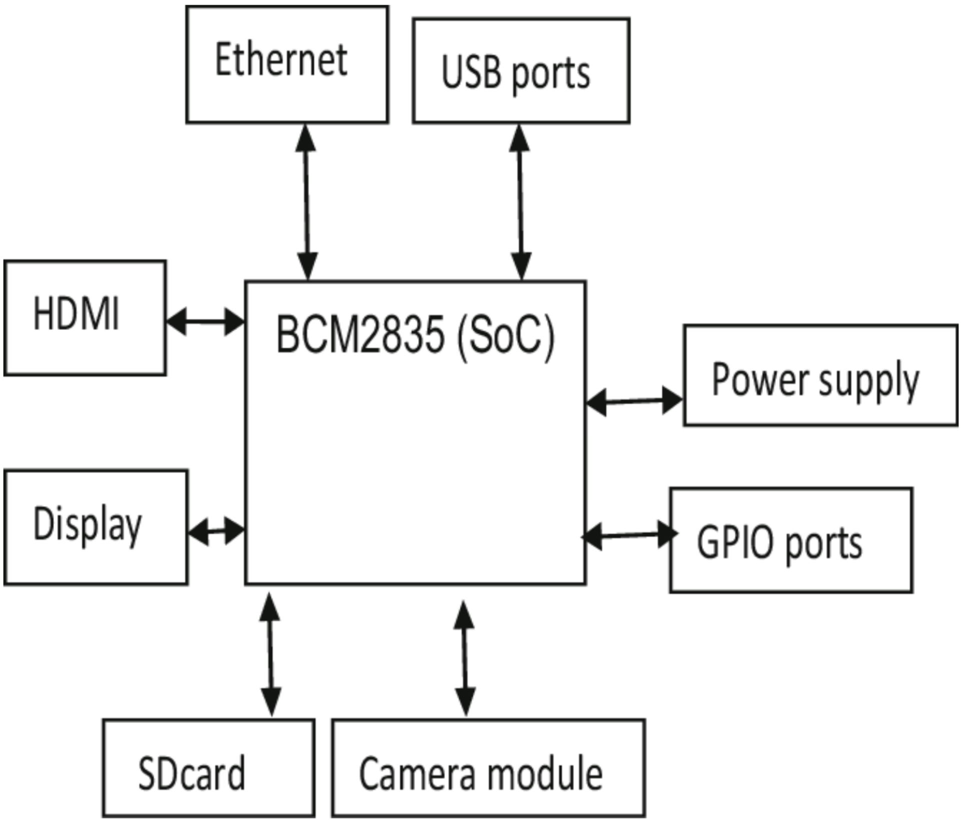
- Program to read the temperature using temperature sensor LM35 for the Arduino Uno Board -

▼ Code

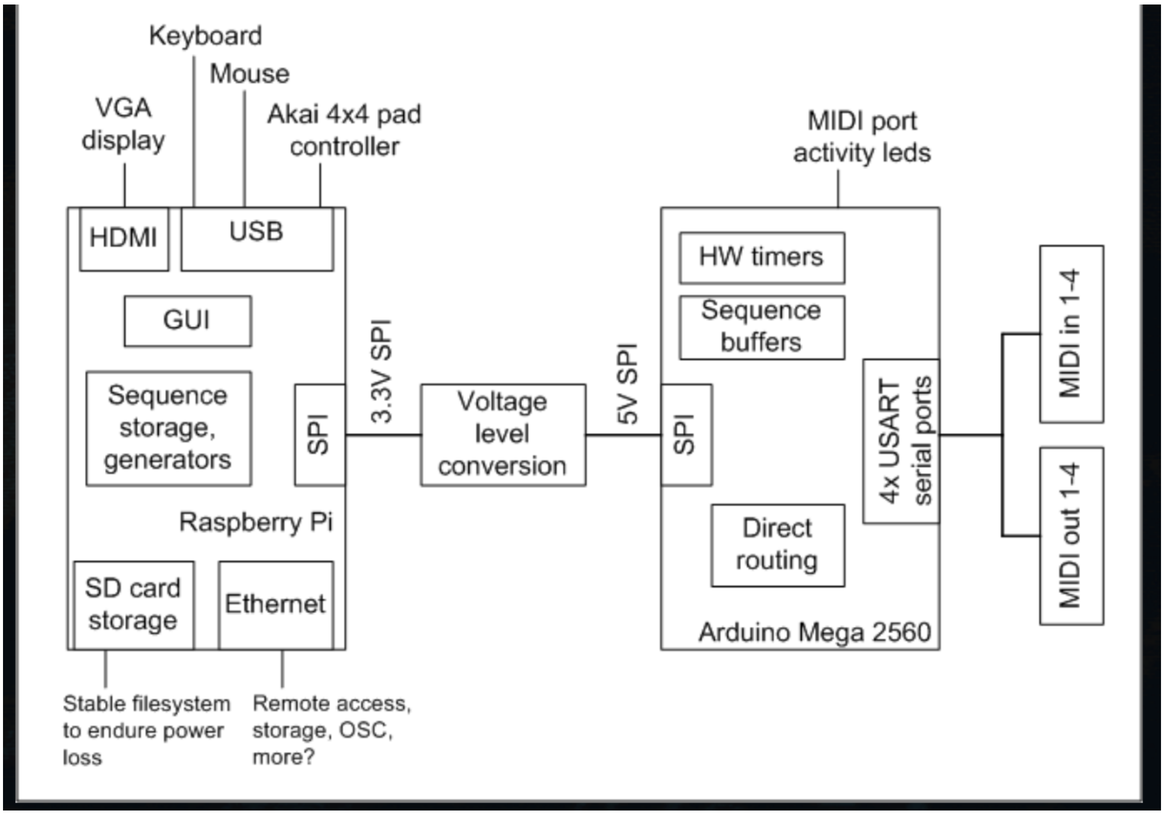
```
// C++ code
//
//include the library for lcd display
#include <LiquidCrystal.h>

//initialise the library with numbers of the interface pins
LiquidCrystal lcd(13,12,5,4,3,2);
int a[]={9,8,11,10};
int len=sizeof(a);

void setup()
{
  //setup the Lcd's number of columns and rows:
  lcd.begin(16,2);
  lcd.setCursor(0,0);
  //Print a message to Lcd
  //lcd.print("Temp C:");
  //Arduino pin number to set the mode of pin
  for(int i=0;i<len;i++){
    pinMode(a[i], OUTPUT);
  }
}

void loop()
{
  double C = temp(analogRead(A5));
  //set //the cursor to column 0, line 1
  //(note: line 1 is the A5nd row, since counting begins with 0)
  lcd.setCursor(0,1);
  //print the temperature
```

```
  lcd.print("place 1: ");
  lcd.print(C);
  glow(int(C));
  delay(2000);

  C = temp(analogRead(A4));
  //set //the cursor to column 0, line 1
  //(note: line 1 is the A5nd row, since counting begins with 0)
  lcd.setCursor(0,1);
  //print the temperature
  lcd.print("place 2: ");
  lcd.print(C);0
  glow(int(C));
  delay(2000);

  C = temp(analogRead(A3));
  //set //the cursor to column 0, line 1
  //(note: line 1 is the A5nd row, since counting begins with 0)
  lcd.setCursor(0,1);
  //print the temperature
  lcd.print("place 3: ");
  lcd.print(C);
  glow(int(C));
  delay(2000);

  C = temp(analogRead(A2));
  //set //the cursor to column 0, line 1
  //(note: line 1 is the A5nd row, since counting begins with 0)
  lcd.setCursor(0,1);
  //print the temperature
  lcd.print("place 4: ");
  lcd.print(C);
  glow(int(C));
  delay(2000);
}

double temp(double x){
  //Serial.print(x):
  //read data /1024 <find percentage>
  // -> convert into voltage
  // -> minus offset
  // -> convert into degree
  return ((((x/1024)*5)-0.5)*100);
}

void glow(int x){
  set_all_low();
  if(x<=30){
    digitalWrite(a[0],1);
    //Write a High value to a digital pin
  }
  else if(x<=60){
    digitalWrite(a[1],1);
    //Write a High value to a digital pin
  }
  else if(x<=90){
    digitalWrite(a[2],1);
    //Write a High value to a digital pin
  }
  else{
    digitalWrite(a[3],1);
    //Write a High value to a digital pin
  }
}

void set_all_low(){
  for(int i=0;i<len;i++){
    digitalWrite(a[i],0);
    //Write a Low value to a digital pin
  }
}
```

- Raspberry PI architecture

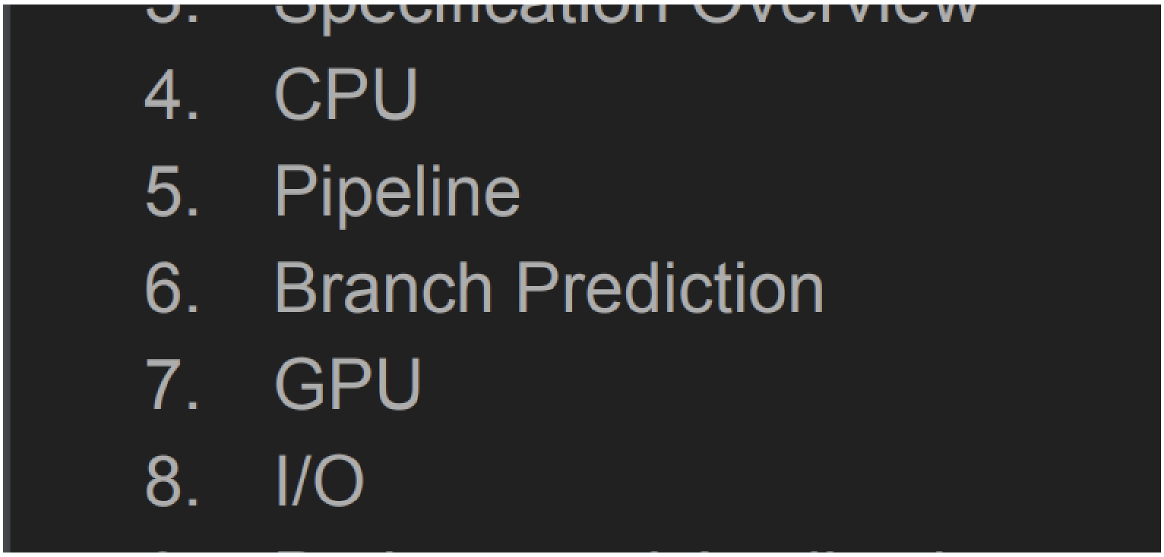


Raspberry pi hardware architecture



- Raspberry pi features -

Memory



▼ unit 5

rfid -

RFID (radio frequency identification)

What is RFID (radio frequency identification)?

RFID (radio frequency identification) is a form of wireless communication that incorporates the use of electromagnetic or electrostatic coupling in the radio frequency portion of the electromagnetic spectrum to uniquely identify an object, animal or person.

How does RFID work?

Every RFID system consists of three components: a scanning antenna, a transceiver and a transponder. When the scanning antenna and transceiver are combined, they are referred to as an RFID reader or interrogator. There are two types of RFID readers -- fixed readers and mobile readers. The RFID reader is a network-connected device that can be portable or permanently attached. It uses radio waves to transmit signals that activate the tag. Once activated, the tag sends a wave back to the antenna, where it is translated into data.

The transponder is in the RFID tag itself. The read range for RFID tags varies based on factors including the type of tag, type of reader, RFID frequency and interference in the surrounding environment or from other RFID tags and readers. Tags that have a stronger power source also have a longer read range.

What are RFID tags and smart labels?

RFID tags are made up of an integrated circuit (IC), an antenna and a substrate. The part of an RFID tag that encodes identifying information is called the RFID inlay.

There are two main types of RFID tags:

- **Active RFID.** An active RFID tag has its own power source, often a battery.
- **Passive RFID.** A passive RFID tag receives its power from the reading antenna, whose electromagnetic wave induces a current in the RFID tag's antenna.

There are also semi-passive RFID tags, meaning a battery runs the circuitry while communication is powered by the RFID reader.

Low-power, embedded non-volatile memory plays an important role in every RFID system. RFID tags typically hold less than 2,000 KB of data, including a unique identifier/serial number. Tags can be read-only or read-write, where data can be added by the reader or existing data overwritten.

The read range for RFID tags varies based on factors including type of tag, type of reader, RFID frequency, and interference in the surrounding environment or from other RFID tags and readers. Active RFID tags have a longer read range than passive RFID tags due to the stronger power source.

smart sensor -

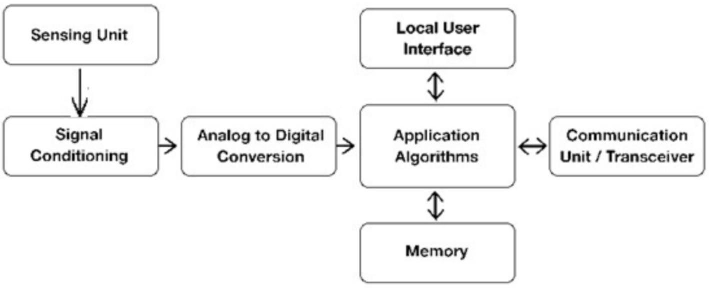
In instrumentation systems, sensors are very essential devices. At present, most of the types of sensors
 are smart. So in these sensors, the sensing elements & electronics are integrated on the same chip. So, the integration of electronics and sensors to make an intelligent sensor is known as a **smart sensor.** This sensor can make some decisions. These sensors have many benefits like higher S/N ratio, fast signal conditioning, auto-calibration, self-testing, high reliability, small physical size, detection & prevention of failure. So, this article discusses an overview of a smart sensor, it’s working, and its applications.

A smart sensor is a device that uses a transducer to gather particular data from a physical environment to perform a predefined & programmed function on the particular type of gathered data then it transmits the data through a networked connection.

The **features of the smart sensor** are; self-identification, digital sensor data, smart calibration & compensation, multi-sensing capacity, sensor communication for configuration of remote & remote monitoring, etc.

Smart Sensor Block Diagram

The block diagram of the smart sensor is shown below. This block diagram includes different blocks like sensing unit, signal conditioning, analog to digital conversion, application algorithms, local user interface, memory, and communication unit or transceiver.



Smart Sensor Block Diagram

Sensing Unit

This unit detects the changes in physical parameters & generates electrical signals equivalent to it. Signal

Conditioning Unit

The signal conditioning unit controls the signal to meet the necessities of next-level operations without losing data.

Analog to Digital Converter

ADC converts the signal from analog to digital format & sends it to the microprocessor.

Local User Interface

The local user interface or LUI is a panel-mounted device used to allow building operators to monitor & control system equipment.

Application Algorithm

The signals from smart sensors reach here & process the received data based on the application programs previously loaded here & generate output signals.

Memory

It is used to store media for saving received & processed data.

Communication Unit

The output signals from the application algorithm or microprocessor are transmitted to the main station through the communication unit. This unit also gets command requirements from the key station to execute specific tasks.

Thinkspeak

Learn More About ThingSpeak

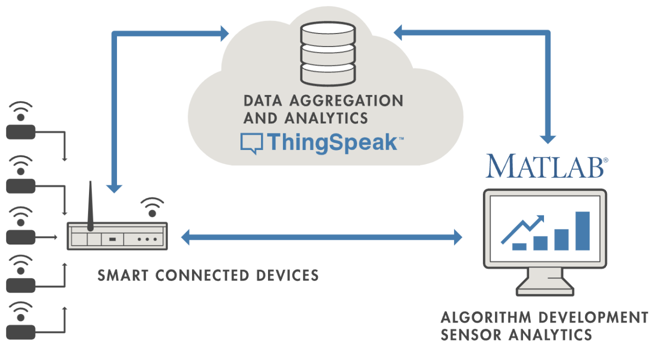
ThingSpeak™ is an IoT analytics platform service that allows you to aggregate, visualize and analyze live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by your devices to ThingSpeak. With the ability to execute MATLAB® code in ThingSpeak you can perform online analysis and processing of the data as it comes in. ThingSpeak is often used for prototyping and proof of concept IoT systems that require analytics.

What is IoT?

Internet of Things (IoT) describes an emerging trend where a large number of embedded devices (things) are connected to the Internet. These connected devices communicate with people and other things and often provide sensor data to cloud storage and cloud computing resources where the data is processed and analyzed to gain important insights. Cheap cloud computing power and increased device connectivity is enabling this trend.

IoT solutions are built for many vertical applications such as environmental monitoring and control, health monitoring, vehicle fleet monitoring, industrial monitoring and control, and home automation.

At a high level, many IoT systems can be described using the diagram below:



ThingSpeak Key Features

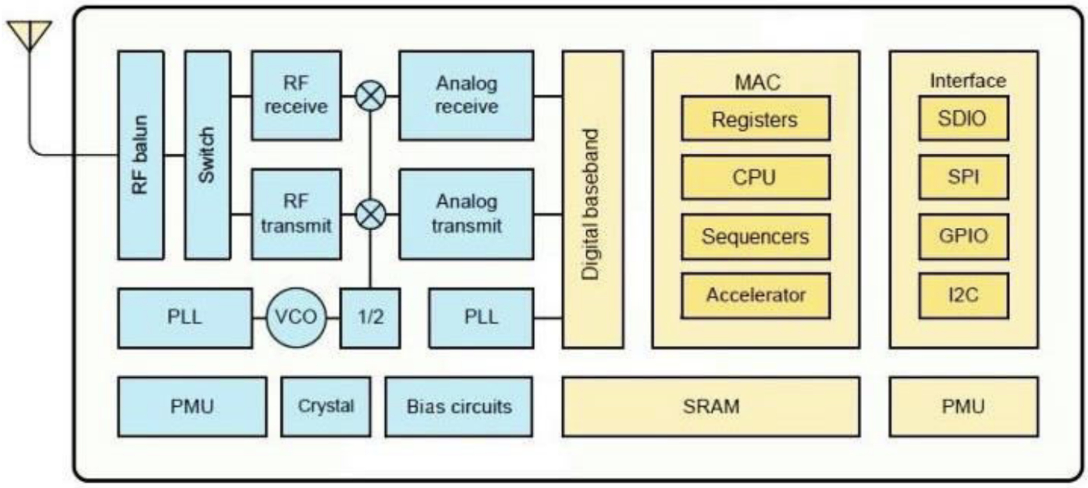
ThingSpeak allows you to aggregate, visualize and analyze live data streams in the cloud. Some of the key capabilities of ThingSpeak include the ability to:

- Easily configure devices to send data to ThingSpeak using popular IoT protocols.
- Visualize your sensor data in real-time.
- Aggregate data on-demand from third-party sources.
- Use the power of MATLAB to make sense of your IoT data.
- Run your IoT analytics automatically based on schedules or events.
- Prototype and build IoT systems without setting up servers or developing web software.
- Automatically act on your data and communicate using third-party services like Twilio® or Twitter®.

To learn how you can collect, analyze and act on your IoT data with ThingSpeak, explore the topics below:

Collect	Analyze	Act
Send sensor data privately to the cloud.	Analyze and visualize your data with MATLAB.	Trigger a reaction.

block diagram of node mcu



ESP8266 block diagram with its main internal functional blocks

Control dc motor python raspberry -

```
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BOARD)

Motor1A = 16
Motor1B = 18
Motor1E = 22

GPIO.setup(Motor1A,GPIO.OUT)
GPIO.setup(Motor1B,GPIO.OUT)
GPIO.setup(Motor1E,GPIO.OUT)

print "Going forwards"
GPIO.output(Motor1A,GPIO.HIGH)
GPIO.output(Motor1B,GPIO.LOW)
GPIO.output(Motor1E,GPIO.HIGH)

sleep(2)

print "Going backwards"
GPIO.output(Motor1A,GPIO.LOW)
GPIO.output(Motor1B,GPIO.HIGH)
GPIO.output(Motor1E,GPIO.HIGH)

sleep(2)

print "Now stop"
GPIO.output(Motor1E,GPIO.LOW)

GPIO.cleanup()
```

- **Room Temperature Measurement with Raspberry Pi**

```
import RPi.GPIO as IO            # calling for header file which helps us use GPIO’s of PI

import time                              # calling for time to provide delays in program

import sys

IO.setwarnings(False)            # do not show any warnings

x=1

b0 =0                                       # integer for storing the delay multiple

b1 =0

b2 =0

b3 =0

b4 =0

b5 =0

b6 =0

b7 =0

IO.setmode (IO.BCM)                # programming the GPIO by BCM pin numbers. (like PIN29 as‘GPIO5’)

IO.setup(4,IO.IN)                        # initialize GPIO Pins as input

IO.setup(17,IO.IN)

IO.setup(27,IO.IN)

IO.setup(22,IO.IN)

IO.setup(5,IO.IN)

IO.setup(6,IO.IN)

IO.setup(13,IO.IN)

IO.setup(19,IO.IN)

while 1:                                              # execute loop forever

    if (IO.input(19) == True):

        time.sleep(0.001)

        if (IO.input(19) == True):

            b7=1                                     # if pin19 is high bit7 is true

    if (IO.input(13) == True):

        time.sleep(0.001)

        if (IO.input(13) == True):

            b6=1                                     # if pin13 is high bit6 is true

    if (IO.input(6) == True):

        time.sleep(0.001)

        if (IO.input(6) == True):

            b5=1                                    # if pin6 is high bit5 is true

    if (IO.input(5) == True):
```

```
    time.sleep(0.001)

    if (IO.input(5) == True):

        b4=1                                   # if pin5 is high bit4 is true

if (IO.input(22) == True):

    time.sleep(0.001)

    if (IO.input(22) == True):

        b3=1                                  # if pin22 is high bit3 is true

if (IO.input(27) == True):

    time.sleep(0.001)

    if (IO.input(27) == True):

        b2=1                                 # if pin27 is high bit2 is true

if (IO.input(17) == True):

    time.sleep(0.001)

    if (IO.input(17) == True):

        b1=1                                  # if pin17 is high bit1 is true

if (IO.input(4) == True):

    time.sleep(0.001)

    if (IO.input(4) == True):

        b0=1                                      # if pin4 is high bit0 is true


x = (1*b0)+(2*b1)                        # representing the bit values from LSB to MSB

x = x+(4*b2)+(8*b3)

x = x+(16*b4)+(32*b5)

x = x+(64*b6)+(128*b7)

x = x/1.275

#temp=100,ref=2000mv,read=255/200=1.275countper10mv or 1.275count for 1degree

print ( x)                                              # print the ADC value

b0=b1=b2=b3=b4=b5=b6=b7=0      # reset the values

time.sleep(0.01)                                 # wait for 10ms
```