A

Project Report On

Personal Info Assistant

Design Engineering-2B (2160001)

BACHELOR OF ENGINEERING
in
ELECTRONICS AND COMMUNICATION ENGINEERING

By

Chaitanya Tejaswi (140080111013)

Under The Guidance of
Prof Kaushal Patel
Professor, ET Department.

ELECTRONICS & COMMUNICATION ENGINEERING
DEPARTMENT
BVM ENGINEERING COLLEGE
GUJARAT TECHNOLOGICAL UNIVERSITY
VALLABH VIDYANAGAR-388120
Academic Year- 2016-17

CERTIFICATE

This is to certify that the project report entitled ***"Personal Info Assistant",*** submitted by ***Chaitanya Tejaswi (140080111013)*** in the subject of the ***Design Engineering-2B (2160001)*** for the *Bachelor of Engineering in Electronics and Communication* of *BVM Engineering College, Vallabh Vidyanagar (Gujarat Technological University)*, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination.

Under The Guidance Of
Prof Kaushal Patel
Professor, ET Department.

ELECTRONICS & COMMUNICATION ENGINEERING
DEPARTMENT
BVM ENGINEERING COLLEGE
GUJARAT TECHNOLOGICAL UNIVERSITY
VALLABH VIDYANAGAR-388120
Academic Year- 2016-17

INDEX

Introduction

What is a Personal Info Assistant (PIA)?

Following the idea of a "Personal Digital Assistant" (PDA), a PIA can be defined as a hardware which runs applications that provide quick reference to lists and processed data through proper links.

Why PIA?

We live in the age of information.

In the entire length of time between waking up to the sound of an alarm set on our branded smartphones and setting the same alarm before going to bed at night, we encounter a wide variety of tasks every day.

Common among these activities is the fact that each of these activities expects us to be informed. Using a washing machine needs us to know how to operate the buttons. Using an air-conditioner needs us to know what buttons to press on the remote in order to get the right setting.

Well, these are simple, aren't they?

Yes, because manufacturers make their products easy to use by hiding their inner features.

What's your response when your washing machine wouldn't run no matter how many buttons you press or your AC won't cool at the right temperature?

We all get irritated, don't we? The best response we have is to call *customer care* or a *repairman*.

Right at that moment something crosses my mind. Given our dependency on these, how little do we know about the things around us!

How much do we know about our 32" LCD Plasma TV; about the 5-speed automatic washing machine; about the 4-star rated AC; about the RO Water Purifier – all of which we operate on a daily basis at our homes?

"I'm not a techie guy", you may say, and you're correct. It's not necessary to knew everything about them, but a working knowledge wouldn't hurt.

This issue is serious for students & professionals.

Not getting the desired output on the CRO no matter how well you have connected the circuits? Does it bother you why it happened?

Been there. Done that.

The more we go out, the more we learn how little we know. But that is no excuse to have no idea about not having any idea about things we use on a daily basis.

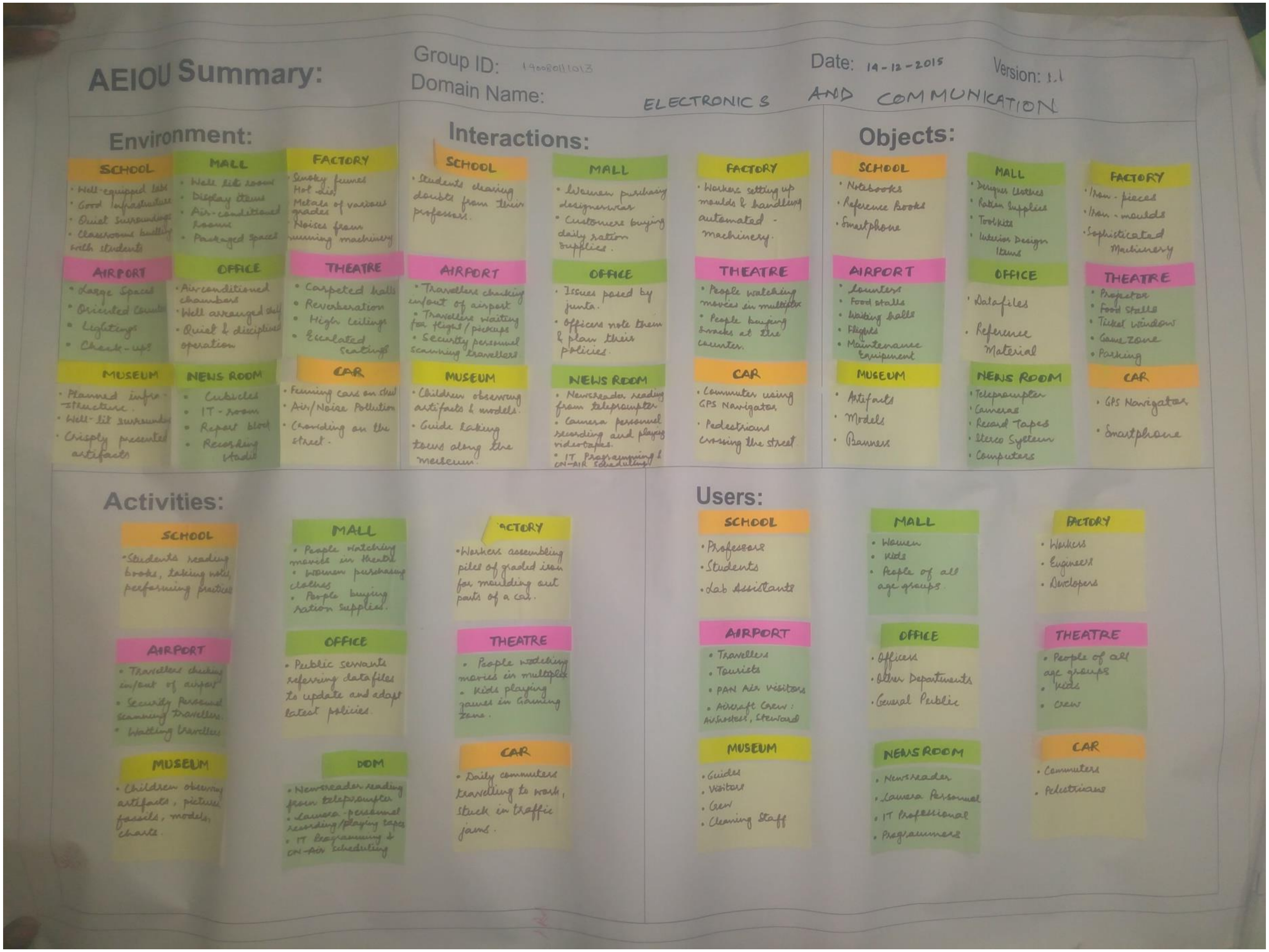
The only problem is – **How would it be done?**

No one has the time or desire to leaf through the pages of a user guide; or to search for authentic documentation on the device.

Well, what if I told you that you do not need to do any of these; the information will reveal itself to you! What would your response be?

Well, this project may do just that.

Observation Record Sheets: AEIOU



AEIOU Summary:

Group ID: 140080111013
Date: 14-12-2015
Version: 1.1
Domain Name: ELECTRONICS AND COMMUNICATION

Environment:

SCHOOL
- Well-equipped labs
- Good Infrastructure
- Quiet surroundings
- Classrooms bustling with students

MALL
- Well lit rooms
- Display items
- Air-conditioned Rooms
- Packaged spaces

FACTORY
- Smoky fumes
- Hot air
- Metals of various grades
- Noises from running machinery

AIRPORT
- Large Spaces
- Oriented Counters
- Lightings
- Check-ups

OFFICE
- Air conditioned chambers
- Well arranged shelf
- Quiet & disciplined operation

THEATRE
- Carpeted halls
- Reverberation
- High ceilings
- Escalated seatings

MUSEUM
- Planned infra-structure.
- Well-lit surroundings
- Crisply presented artifacts

NEWS ROOM
- Cubicles
- IT-room
- Report block
- Recording studio

CAR
- Fuming cars on the street
- Air/Noise Pollution
- Crowding on the street.

Interactions:

SCHOOL
- Students clearing doubts from their professors.

MALL
- Women purchasing designerwear
- Customers buying daily ration supplies.

FACTORY
- Workers setting up moulds & handling automated - machinery.

AIRPORT
- Travellers checking in/out of airport
- Travellers waiting for flight/pickups
- Security personnel scanning travellers

OFFICE
- Issues posed by junta.
- Officers note them & plan their policies.

THEATRE
- People watching movies in multiplex
- People buying snacks at the counter.

MUSEUM
- Children observing artifacts & models.
- Guide taking tours along the museum.

NEWS ROOM
- Newsreader reading from teleprompter
- Camera personnel recording and playing videotapes.
- IT Programming & ON-AIR Scheduling

CAR
- Commuter using GPS Navigator
- Pedestrians crossing the street

Objects:

SCHOOL
- Notebooks
- Reference Books
- Smartphone

MALL
- Designer clothes
- Ration Supplies
- Toolkits
- Interior Design Items

FACTORY
- Iron-pieces
- Iron-moulds
- Sophisticated Machinery

AIRPORT
- Counters
- Food stalls
- Waiting halls
- Flights
- Maintenance Equipment

OFFICE
- Datafiles
- Reference Material

THEATRE
- Projector
- Food Stalls
- Ticket window
- Game Zone
- Parking

MUSEUM
- Artifacts
- Models
- Banners

NEWS ROOM
- Teleprompter
- Cameras
- Record Tapes
- Stereo System
- Computers

CAR
- GPS Navigator
- Smartphone

Activities:

SCHOOL
- Students reading books, taking notes, performing practicals

MALL
- People watching movies in theatre
- Women purchasing clothes
- People buying ration supplies.

FACTORY
- Workers assembling piles of graded iron for moulding out parts of a car.

AIRPORT
- Travellers checking in/out of airport
- Security Personnel scanning travellers.
- Waiting travellers

OFFICE
- Public servants referring datafiles to update and adapt latest policies.

THEATRE
- People watching movies in multiplex
- Kids playing games in Gaming Zone.

MUSEUM
- Children observing artifacts, pictures, fossils, models, charts.

DOM
- Newsreader reading from teleprompter
- Camera-personnel recording/playing tapes
- IT Programming & ON-Air Scheduling

CAR
- Daily commuters travelling to work, stuck in traffic jams.

Users:

SCHOOL
- Professors
- Students
- Lab Assistants

MALL
- Women
- Kids
- People of all age groups.

FACTORY
- Workers
- Engineers
- Developers

AIRPORT
- Travellers
- Tourists
- PAN Air visitors
- Aircraft Crew: Airhostess, Steward

OFFICE
- Officers
- Other Departments
- General Public

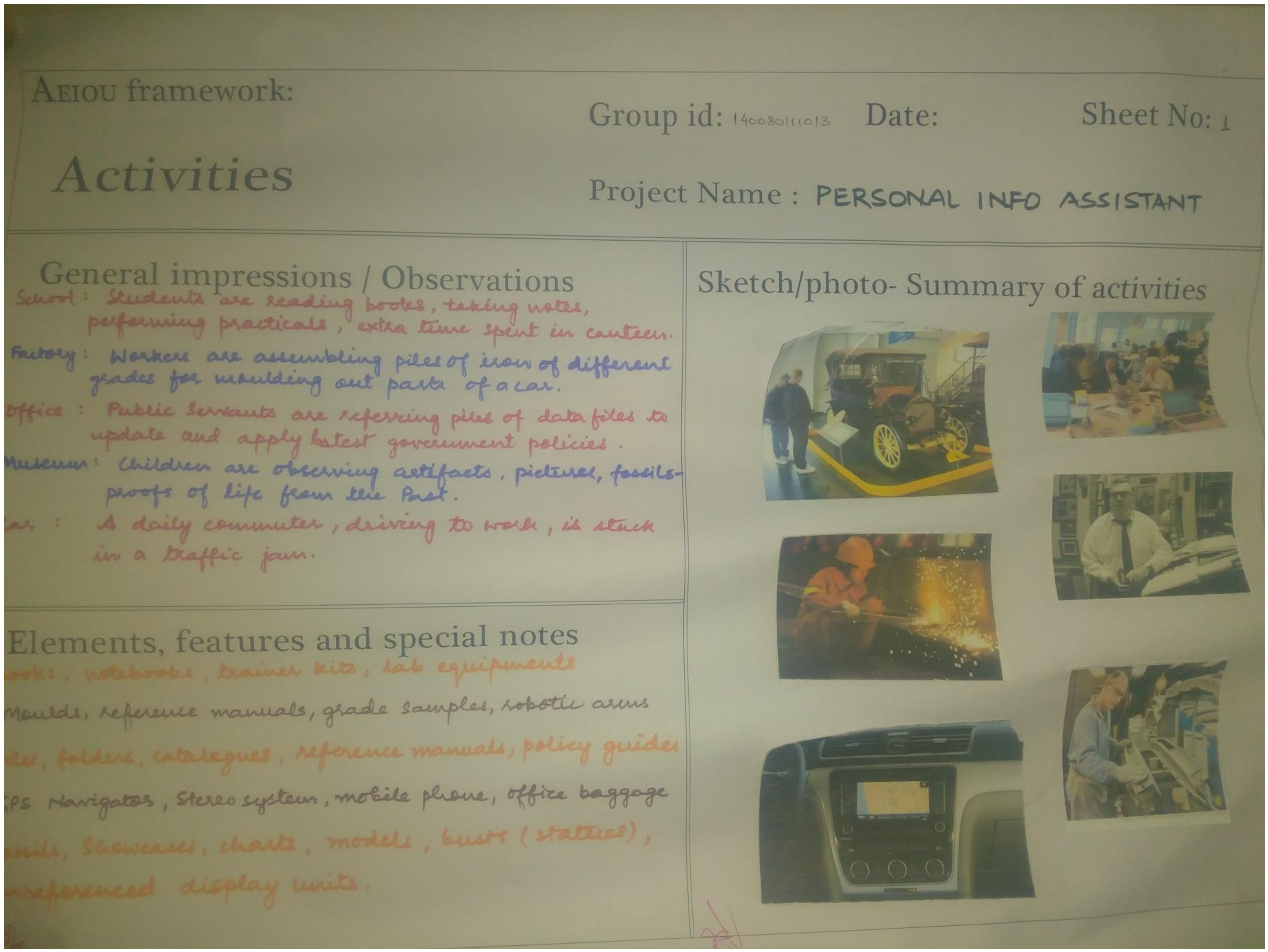
THEATRE
- People of all age groups
- Kids
- Crew

MUSEUM
- Guides
- Visitors
- Crew
- Cleaning Staff

NEWSROOM
- Newsreader
- Camera Personnel
- IT Professional
- Programmers

CAR
- Commuters
- Pedestrians

Activities



AEIOU framework:

Group id: 140080111013 Date: Sheet No: 1

Activities

Project Name : PERSONAL INFO ASSISTANT

General impressions / Observations

School : Students are reading books, taking notes, performing practicals, extra time spent in canteen.

Factory : Workers are assembling piles of iron of different grades for moulding out parts of a car.

Office : Public Servants are referring piles of data files to update and apply latest government policies.

Museum : Children are observing artifacts, pictures, fossils-proofs of life from the Past.

ar : A daily commuter, driving to work, is stuck in a traffic jam.

Sketch/photo- Summary of activities

Elements, features and special notes

ooks, notebooks, trainer kits, lab equipments

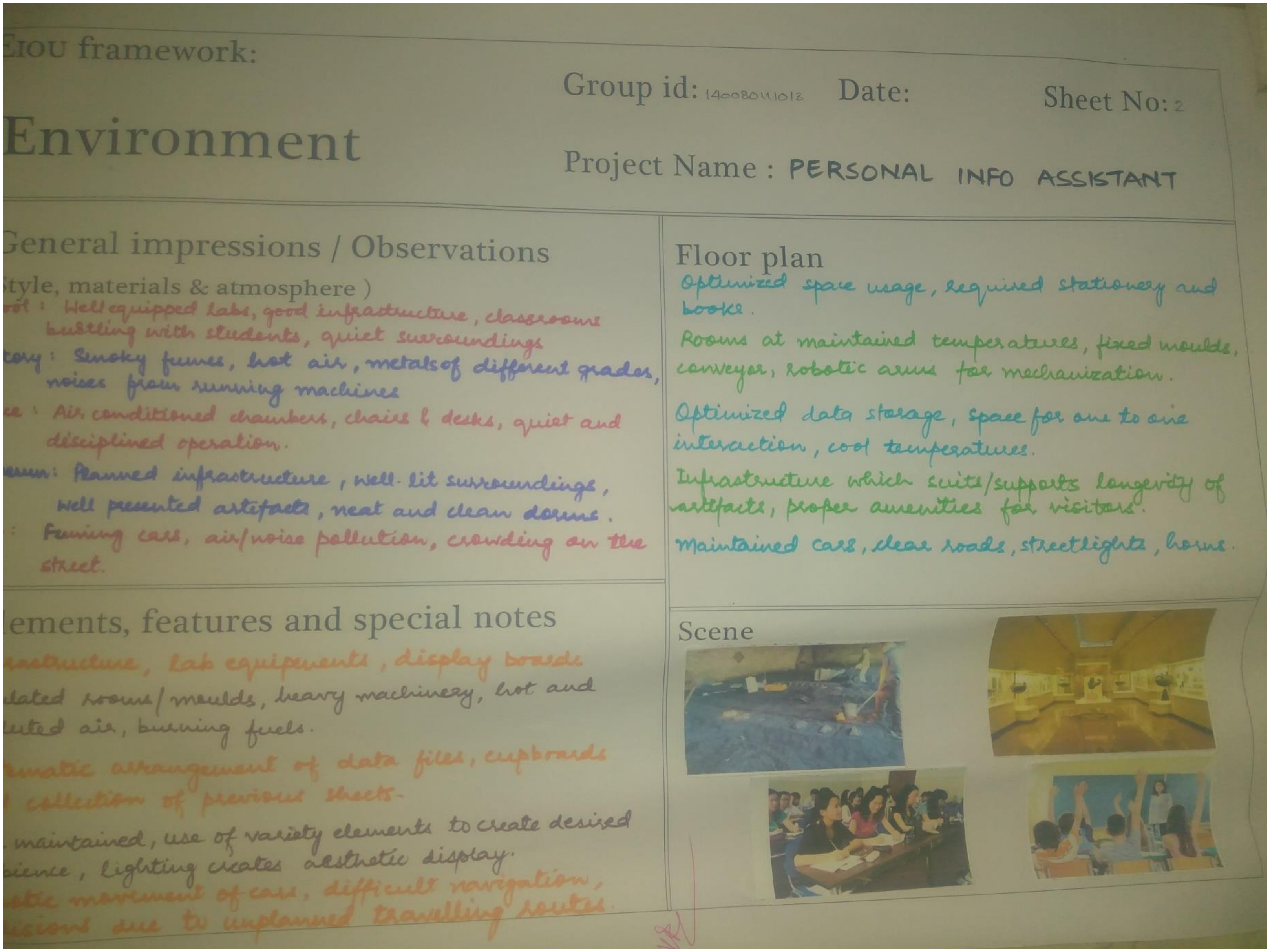
Moulds, reference manuals, grade samples, robotic arms

iles, folders, catalogues, reference manuals, policy guides

PS Navigators, Stereo system, mobile phone, office baggage

ssils, Showcases, charts, models, busts (statues), nreferenced display units.

Environment



EIOU framework:

Environment

Group id: 140080111012 Date: Sheet No: 2

Project Name : PERSONAL INFO ASSISTANT

General impressions / Observations

Style, materials & atmosphere)

ool : Well equipped labs, good infrastructure, classrooms bustling with students, quiet surroundings

tory : Smoky fumes, hot air, metals of different grades, noises from running machines

ce : Air conditioned chambers, chairs & desks, quiet and disciplined operation.

eum: Planned infrastructure, well-lit surroundings, well presented artifacts, neat and clean dorms.

: Fuming cars, air/noise pollution, crowding on the street.

Floor plan

Optimized space usage, required stationary and books.

Rooms at maintained temperatures, fixed moulds, conveyor, robotic arms for mechanization.

Optimized data storage, space for one to one interaction, cool temperatures.

Infrastructure which suits/supports longevity of artifacts, proper amenities for visitors.

Maintained cars, clear roads, streetlights, horns.

lements, features and special notes

astructure, lab equipements, display boards

lated rooms/moulds, heavy machinery, hot and luted air, burning fuels.

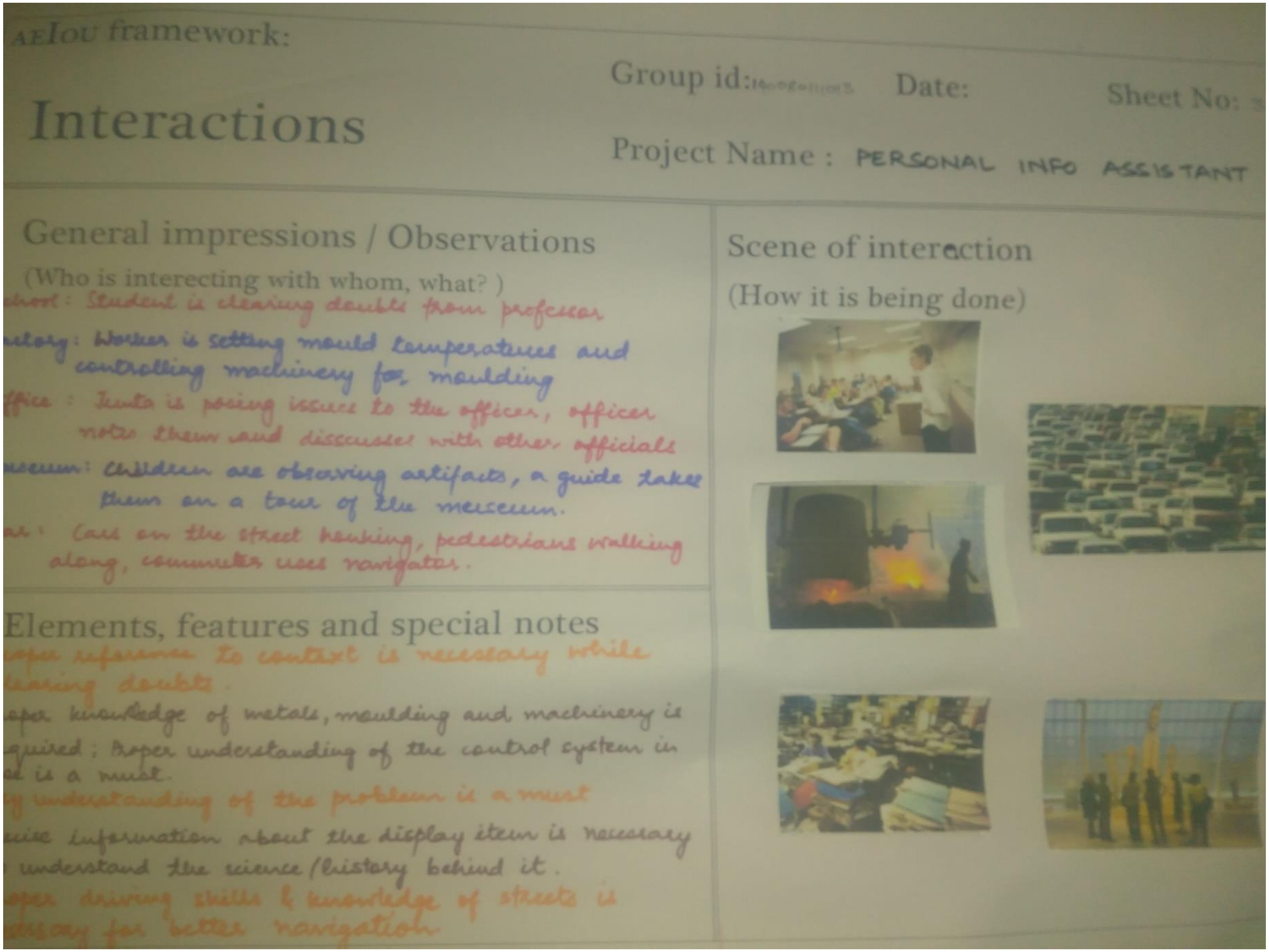
ematic arrangement of data files, cupboards collection of previous sheets.

maintained, use of variety elements to create desired bience, lighting creates aesthetic display.

atic movement of cars, difficult navigation, isions due to unplanned travelling routes.

Scene

Interactions



AEIOU framework:

Interactions

Group id: [illegible] Date: Sheet No: 3

Project Name : PERSONAL INFO ASSISTANT

General impressions / Observations

(Who is interecting with whom, what?)

hool: Student is clearing doubts from professor

ctory: Worker is setting mould temperatures and controlling machinery for moulding

ffice : Junta is posing issues to the officer, officer notes them and disscusses with other officials

useum: Children are observing artifacts, a guide takes them on a tour of the museum.

ar: Cars on the street honking, pedestrians walking along, commuter uses navigator.

Scene of interaction

(How it is being done)

Elements, features and special notes

oper reference to context is necessary while earing doubts.

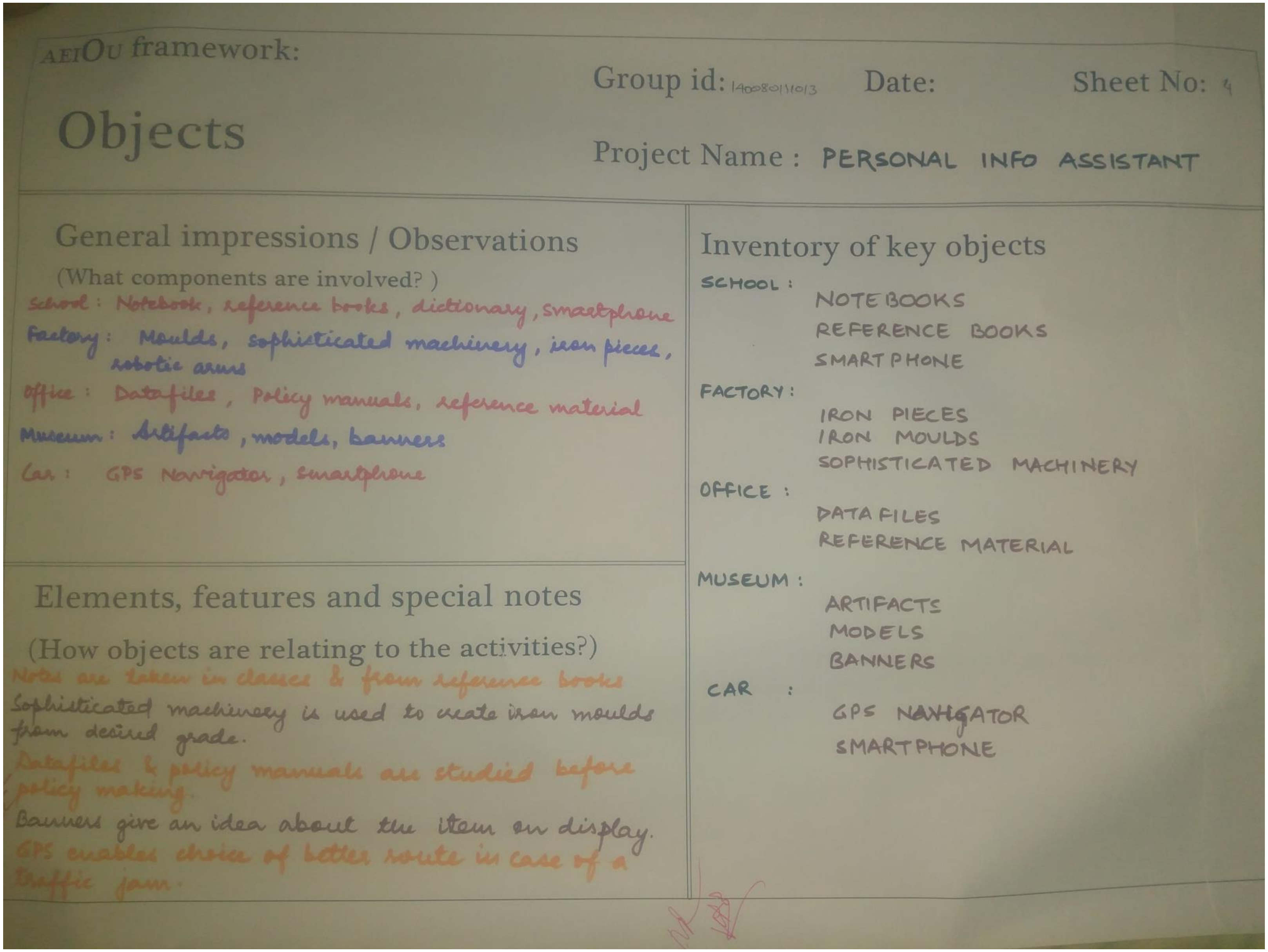
oper knowledge of metals, moulding and machinery is quired; Proper understanding of the control system in d is a must.

y understanding of the problem is a must

ecise information about the display item is necessary understand the science/history behind it.

oper driving skills & knowledge of streets is cessary for better navigation

Objects



AEIOU framework:

Objects

Group id: 140080111013 Date: Sheet No: 4

Project Name : PERSONAL INFO ASSISTANT

General impressions / Observations

(What components are involved?)

School : Notebook, reference books, dictionary, smartphone

Factory : Moulds, sophisticated machinery, iron pieces, robotic arms

Office : Datafiles, Policy manuals, reference material

Museum : Artifacts, models, banners

Car : GPS Navigator, smartphone

Elements, features and special notes

(How objects are relating to the activities?)

Notes are taken in classes & from reference books

Sophisticated machinery is used to create iron moulds from desired grade.

Datafiles & policy manuals are studied before policy making.

Banners give an idea about the item on display.

GPS enables choice of better route in case of a traffic jam.

Inventory of key objects

SCHOOL :
- NOTEBOOKS
- REFERENCE BOOKS
- SMARTPHONE

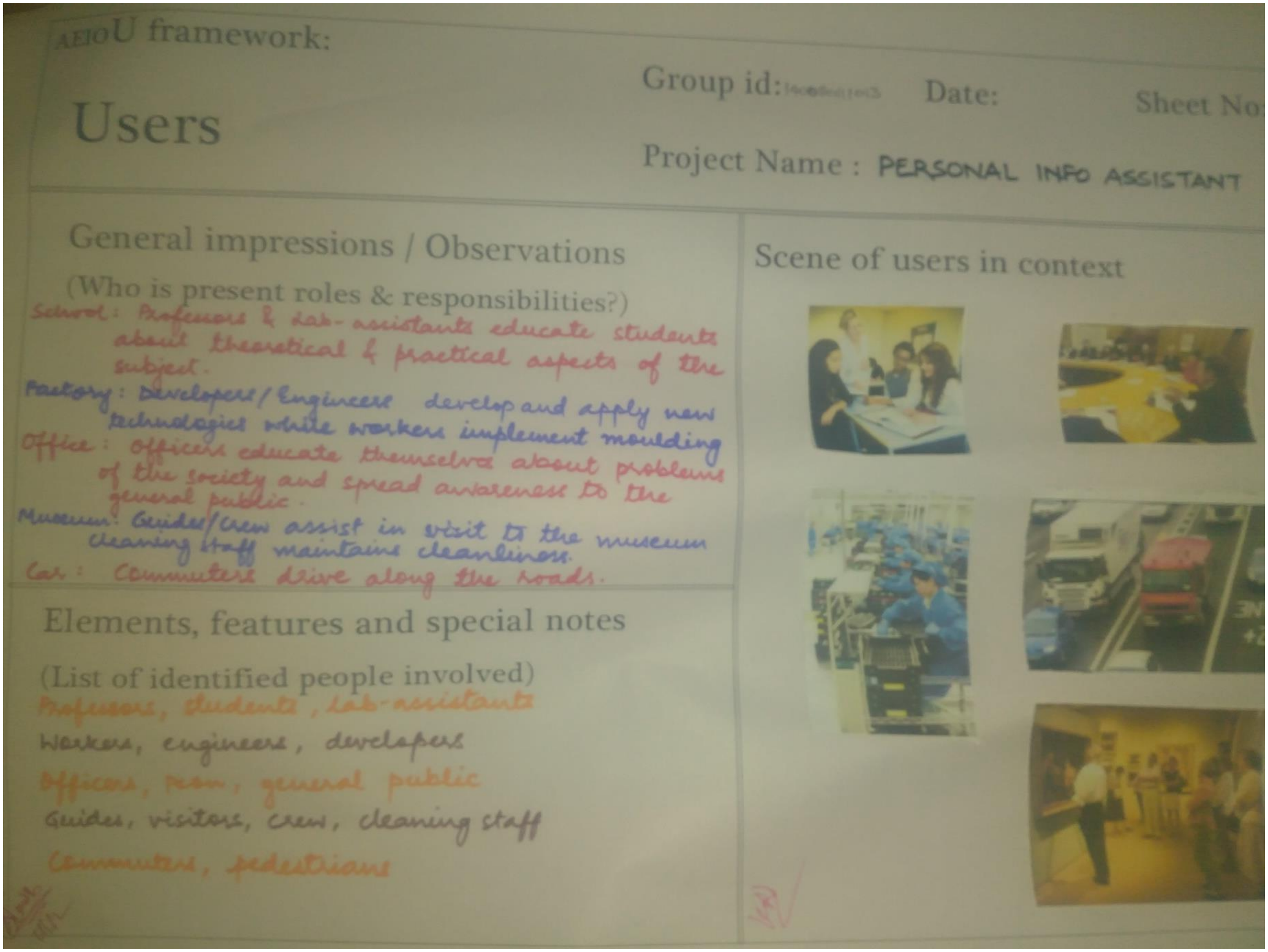
FACTORY :
- IRON PIECES
- IRON MOULDS
- SOPHISTICATED MACHINERY

OFFICE :
- DATA FILES
- REFERENCE MATERIAL

MUSEUM :
- ARTIFACTS
- MODELS
- BANNERS

CAR :
- GPS NAVIGATOR
- SMARTPHONE

Users



AEIOU framework:

Users

Group id: [illegible] Date: Sheet No

Project Name : PERSONAL INFO ASSISTANT

General impressions / Observations

(Who is present roles & responsibilities?)

School : Professors & lab-assistants educate students about theoretical & practical aspects of the subject.

Factory : Developers / Engineers develop and apply new technologies while workers implement moulding

Office : officers educate themselves about problems of the society and spread awareness to the general public.

Museum : Guides/crew assist in visit to the museum cleaning staff maintains cleanliness.

Car : Commuters drive along the roads.

Elements, features and special notes

(List of identified people involved)

Professors, students, lab-assistants

Workers, engineers, developers

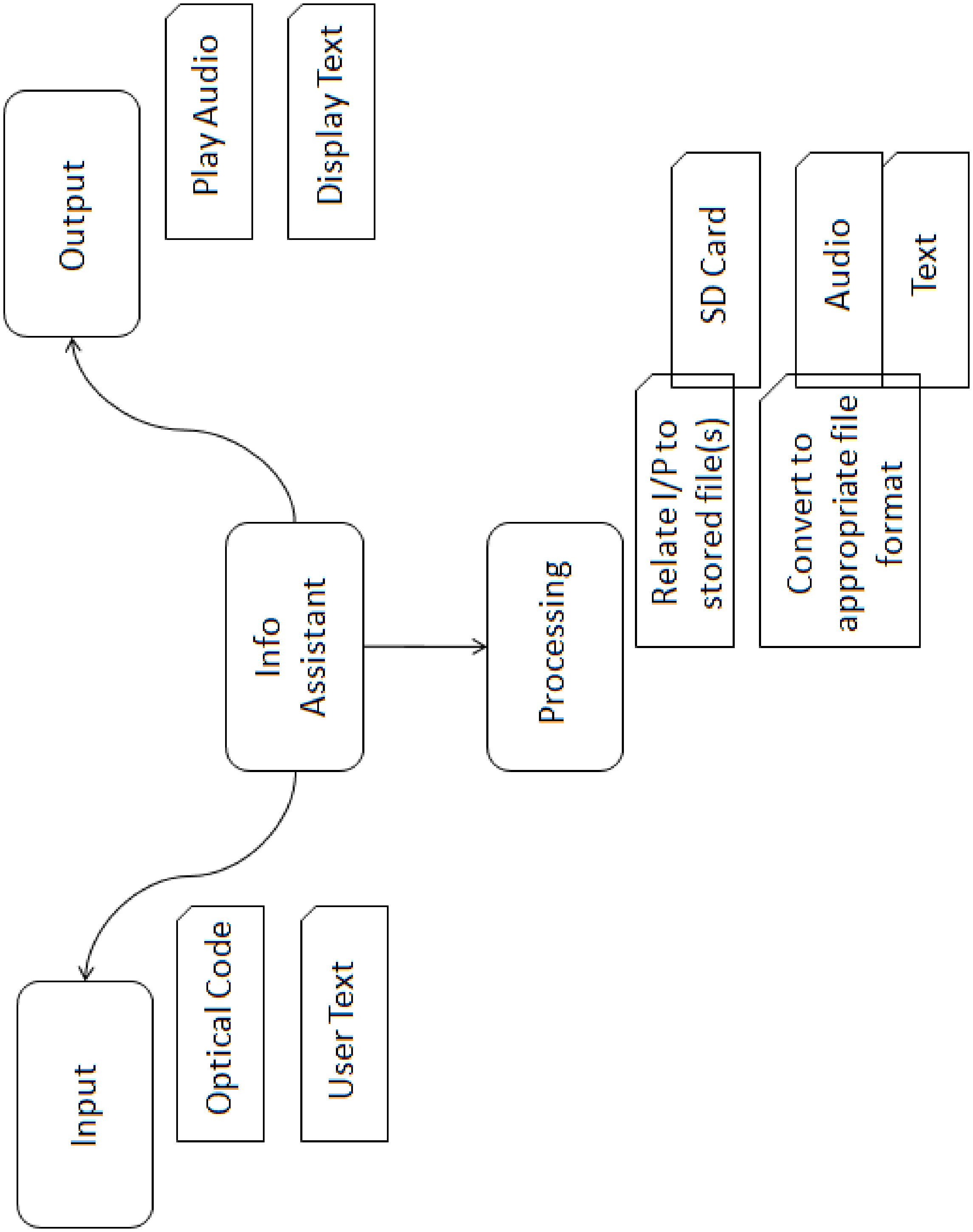
Officers, peon, general public

Guides, visitors, crew, cleaning staff

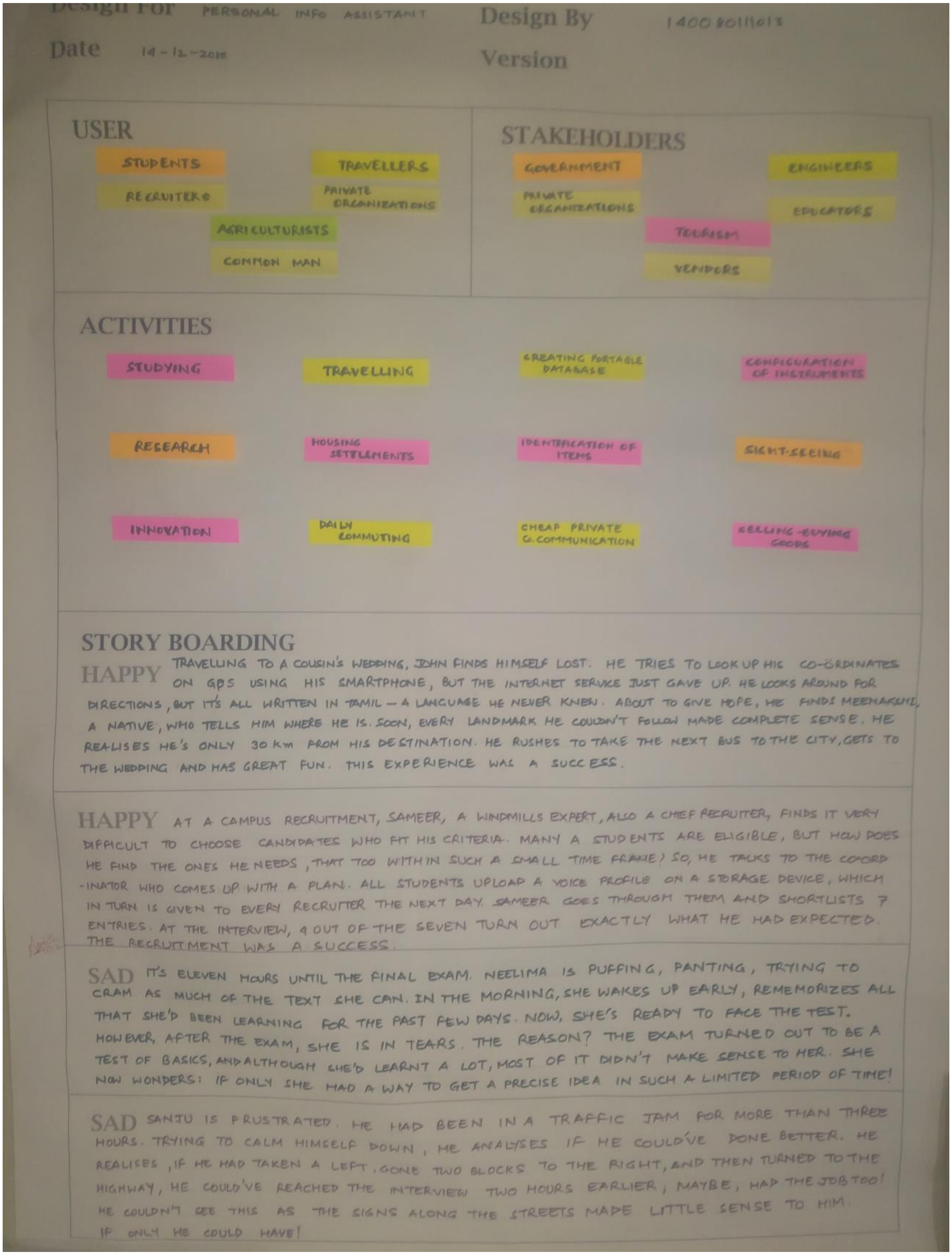
Commuters, pedestrians

Scene of users in context

Observation Record Sheets: MindMap



Canvas: Empathy



Design For PERSONAL INFO ASSISTANT

Design By 1400 8011013

Date 14-12-2018

Version

USER

- STUDENTS
- TRAVELLERS
- RECRUITERS
- PRIVATE ORGANIZATIONS
- AGRICULTURISTS
- COMMON MAN

STAKEHOLDERS

- GOVERNMENT
- ENGINEERS
- PRIVATE ORGANIZATIONS
- EDUCATORS
- TOURISM
- VENDORS

ACTIVITIES

- STUDYING
- TRAVELLING
- CREATING PORTABLE DATABASE
- CONFIGURATION OF INSTRUMENTS
- RESEARCH
- HOUSING SETTLEMENTS
- IDENTIFICATION OF ITEMS
- SIGHT-SEEING
- INNOVATION
- DAILY COMMUTING
- CHEAP PRIVATE & COMMUNICATION
- SELLING-BUYING GOODS

STORY BOARDING

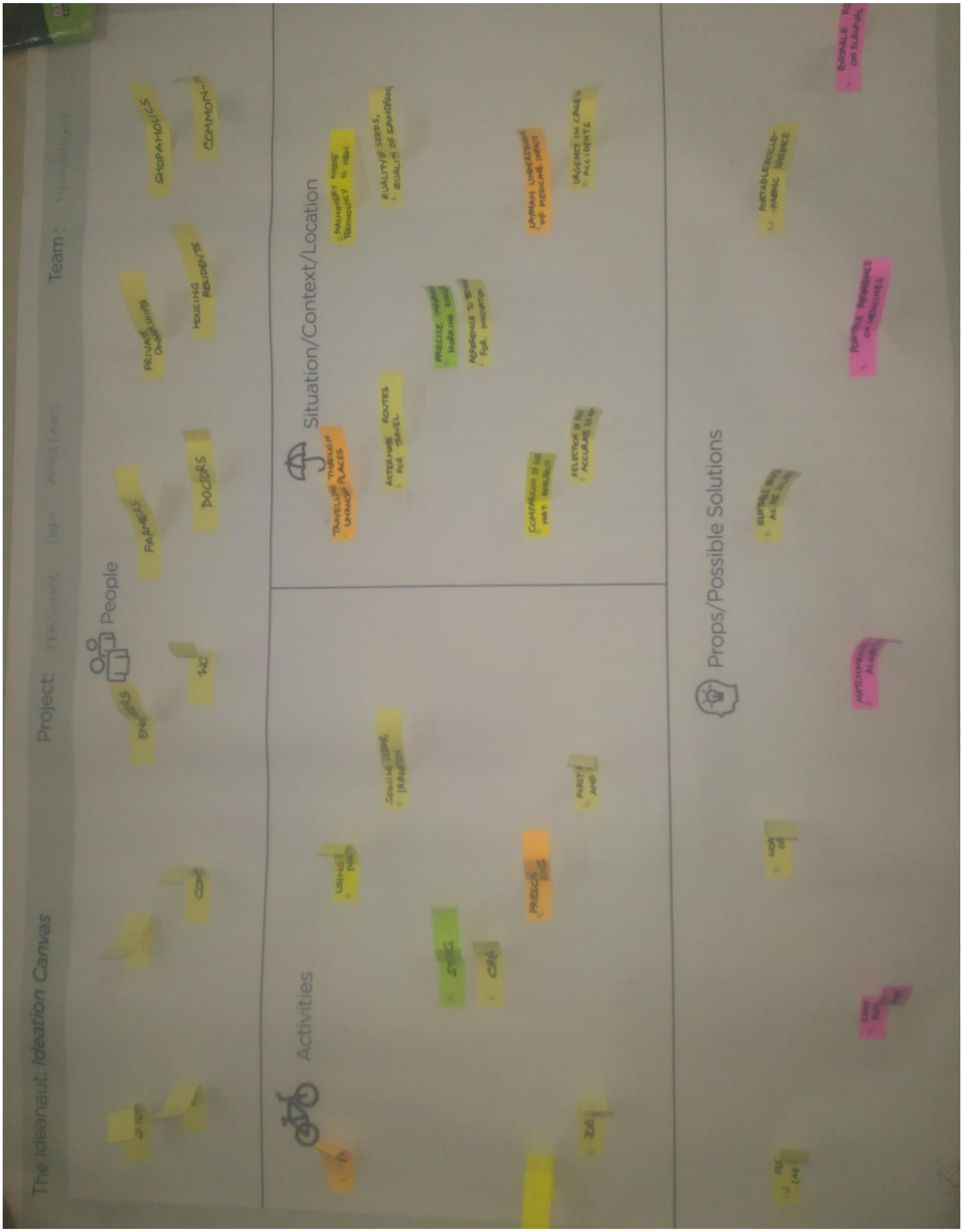
HAPPY TRAVELLING TO A COUSIN'S WEDDING, JOHN FINDS HIMSELF LOST. HE TRIES TO LOOK UP HIS CO-ÖRDINATES ON GPS USING HIS SMARTPHONE, BUT THE INTERNET SERVICE JUST GAVE UP. HE LOOKS AROUND FOR DIRECTIONS, BUT IT'S ALL WRITTEN IN TAMIL — A LANGUAGE HE NEVER KNEW. ABOUT TO GIVE HOPE, HE FINDS MEENAKSHI, A NATIVE, WHO TELLS HIM WHERE HE IS. SOON, EVERY LANDMARK HE COULDN'T FOLLOW MADE COMPLETE SENSE. HE REALISES HE'S ONLY 30 km FROM HIS DESTINATION. HE RUSHES TO TAKE THE NEXT BUS TO THE CITY, GETS TO THE WEDDING AND HAS GREAT FUN. THIS EXPERIENCE WAS A SUCCESS.

HAPPY AT A CAMPUS RECRUITMENT, SAMEER, A WINDMILLS EXPERT, ALSO A CHIEF RECRUITER, FINDS IT VERY DIFFICULT TO CHOOSE CANDIDATES WHO FIT HIS CRITERIA. MANY A STUDENTS ARE ELIGIBLE, BUT HOW DOES HE FIND THE ONES HE NEEDS, THAT TOO WITHIN SUCH A SMALL TIME FRAME? SO, HE TALKS TO THE CO-ORDINATOR WHO COMES UP WITH A PLAN. ALL STUDENTS UPLOAD A VOICE PROFILE ON A STORAGE DEVICE, WHICH IN TURN IS GIVEN TO EVERY RECRUITER THE NEXT DAY. SAMEER GOES THROUGH THEM AND SHORTLISTS 7 ENTRIES. AT THE INTERVIEW, 4 OUT OF THE SEVEN TURN OUT EXACTLY WHAT HE HAD EXPECTED. THE RECRUITMENT WAS A SUCCESS.

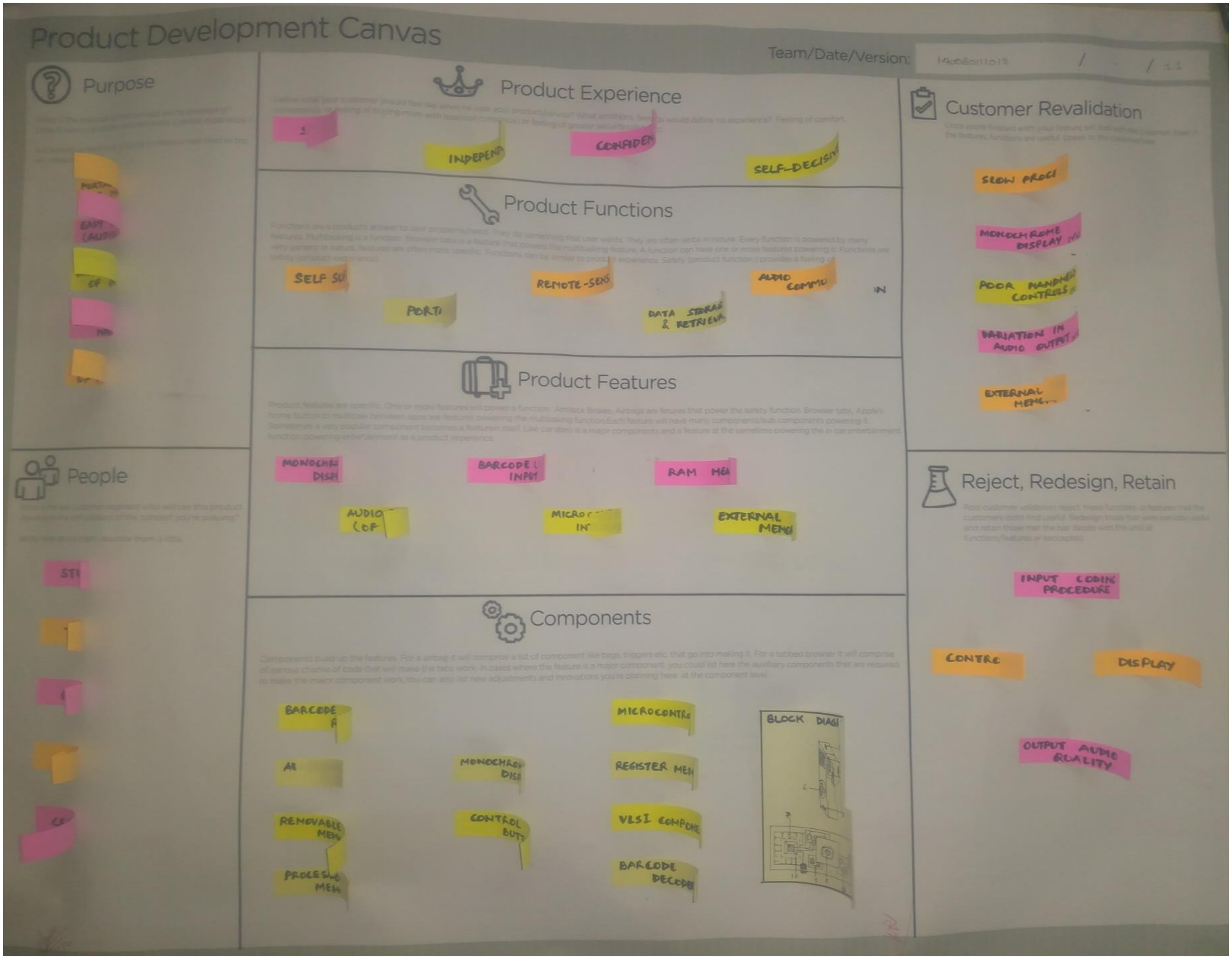
SAD IT'S ELEVEN HOURS UNTIL THE FINAL EXAM. NEELIMA IS PUFFING, PANTING, TRYING TO CRAM AS MUCH OF THE TEXT SHE CAN. IN THE MORNING, SHE WAKES UP EARLY, REMEMORIZES ALL THAT SHE'D BEEN LEARNING FOR THE PAST FEW DAYS. NOW, SHE'S READY TO FACE THE TEST. HOWEVER, AFTER THE EXAM, SHE IS IN TEARS. THE REASON? THE EXAM TURNED OUT TO BE A TEST OF BASICS, AND ALTHOUGH SHE'D LEARNT A LOT, MOST OF IT DIDN'T MAKE SENSE TO HER. SHE NOW WONDERS: IF ONLY SHE HAD A WAY TO GET A PRECISE IDEA IN SUCH A LIMITED PERIOD OF TIME!

SAD SANJU IS FRUSTRATED. HE HAD BEEN IN A TRAFFIC JAM FOR MORE THAN THREE HOURS. TRYING TO CALM HIMSELF DOWN, HE ANALYSES IF HE COULD'VE DONE BETTER. HE REALISES, IF HE HAD TAKEN A LEFT, GONE TWO BLOCKS TO THE RIGHT, AND THEN TURNED TO THE HIGHWAY, HE COULD'VE REACHED THE INTERVIEW TWO HOURS EARLIER; MAYBE, HAD THE JOB TOO! HE COULDN'T SEE THIS AS THE SIGNS ALONG THE STREETS MADE LITTLE SENSE TO HIM. IF ONLY HE COULD HAVE!

Canvas: Ideation


The Ideanaut: Ideation Canvas
Project:
Team:
People
FARMERS
DOCTORS
PRIVATE OWNERSHIPS
HOUSING RESIDENTS
SHOPAHOLICS
COMMON-
Activities
SOWING SEEDS, IRRIGATION
FIRST AID
Situation/Context/Location
TRAVELLING THROUGH UNKNOWN PLACES
ALTERNATE ROUTES FOR TRAVEL
MACHINERY INSIDE TECHNOLOGY IS NEW
QUALITY SEEDS, QUALITY OF LANDSCAPE
LAYMAN UNDERSTANDING OF MEDICINE INTAKE
URGENCY IN CASES OF ACCIDENTS
Props/Possible Solutions
PORTABLE PANIC SENSOR

Canvas: Product Development



Prior Art Search

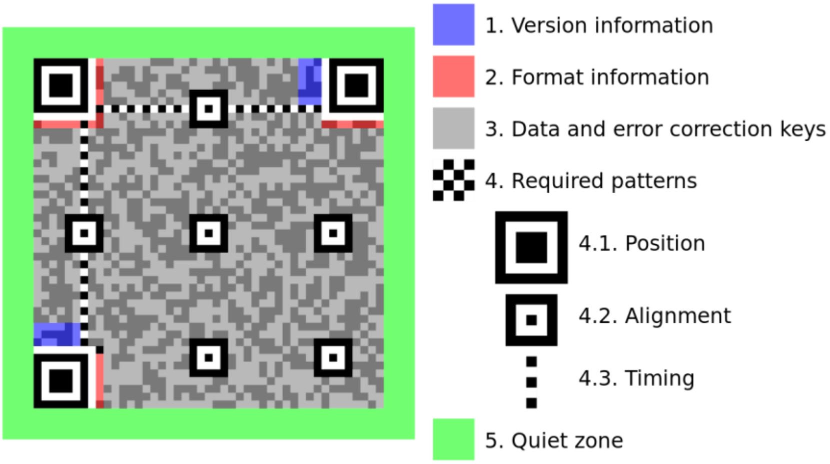
QR-Code

- Summary of QR Code

[US-5726435: “Optically readable two-dimensional code and method and apparatus using the same”]
[US-6997384B2: “Method for displaying and reading information code for commercial transaction”]

Summary of QR Code

Quick Response (QR) code is the trademark for a type of matrix barcode (or two-dimensional barcode) first designed for the automotive industry in Japan.



Structure of a QR code, highlighting functional elements

Design:

1. Unlike the 1D barcodes that were designed to be mechanically scanned by a narrow beam of light, a QR code is detected by a 2D digital image sensor and then digitally analyzed by a programmed processor.
2. The processor locates the three distinctive squares at the corners of the QR code image, using a smaller square (or multiple squares) near the fourth corner to normalize the image for size, orientation, and angle of viewing.
3. The small dots in the QR code are then converted to binary numbers and validated with an error-correcting algorithm.

Storage:

The amount of data that can be stored in the QR code symbol depends on the datatype, version and error correction level.

Maximum character storage capacity (40-L)

Input Mode	Max. Characters	bits/char	Valid characters	Default Encoding
Numeric only	7,089	3⅓	0–9	
Alphanumeric	4,296	5½	0–9, A–Z (upper-case only), space, $, %, *, +, -,., /, :	
Binary/byte	2,953	8		ISO 8859-1
Kanji/kana	1,817	13		Shift JIS X 0208

Error Correction:

1. A Code word is 8-bit long and uses the “Reed–Solomon” error correction algorithm with 4 error correction levels. Higher the error correction level, lesser the storage capacity.

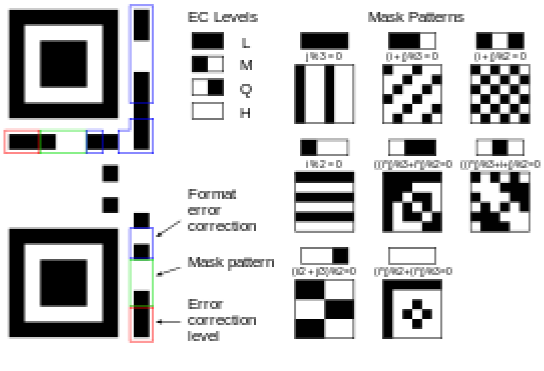
Approximate error correction capabilities

Level	Max. Characters
L (Low)	7% of codewords can be restored.
M (Medium)	15% of codewords can be restored.
Q (Quartile)	25% of codewords can be restored.
H (High)	30% of codewords can be restored.

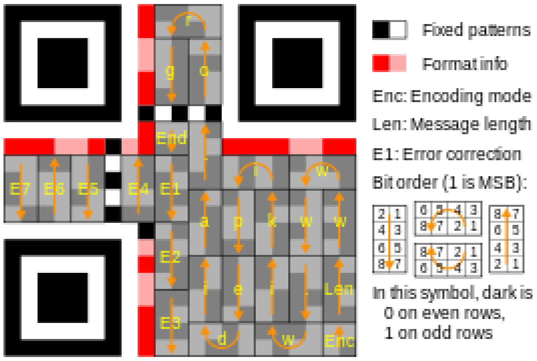
2. In larger QR symbols, the message is broken up into several Reed–Solomon code blocks. The block size is chosen so that at most 15 errors can be corrected in each block; this limits the complexity of the decoding algorithm. The code blocks are then interleaved together, making it less likely that localized damage to a QR symbol will overwhelm the capacity of any single block.
3. Due to error correction, it is possible to create artistic QR codes that still scan correctly, but contain intentional errors to make them more readable or attractive to the human eye, as well as to incorporate colors, logos, and other features into the QR code block.

Encoding:

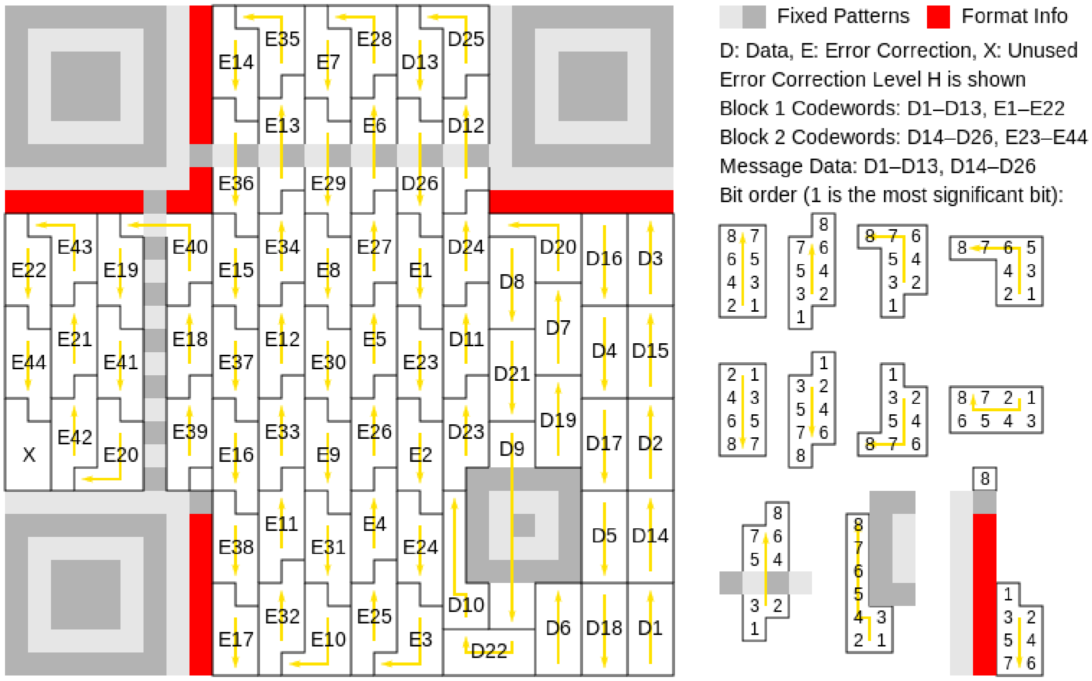
1. The format information records two things: the error correction level and the mask pattern used for the symbol.
2. Masking is used to break up patterns in the data area that might confuse a scanner, such as large blank areas or misleading features that look like the locator marks. The mask patterns are defined on a grid that is repeated as necessary to cover the whole symbol. Modules corresponding to the dark areas of the mask are inverted. The format information is protected from errors with a BCH code, and two complete copies are included in each QR symbol.
3. The message dataset is placed from right to left in a zigzag pattern, as shown below. In larger symbols, this is complicated by the presence of the alignment patterns and the use of multiple interleaved error-correction blocks.



Meaning of format information



Message placement within a QR symbol



⬜ Fixed Patterns 🟥 Format Info

D: Data, E: Error Correction, X: Unused

Error Correction Level H is shown

Block 1 Codewords: D1–D13, E1–E22

Block 2 Codewords: D14–D26, E23–E44

Message Data: D1–D13, D14–D26

Bit order (1 is the most significant bit):

Larger symbol illustrating interleaved blocks

4. The general structure of a QR encoding is as a sequence of 4 bit indicators with payload length dependent on the indicator mode (e.g. byte encoding payload length is dependent on the first byte).

Encoding modes

Mode Indicator	**Description**	**Typical Structure '[type : sizes in bits]'**
0001	Numeric	[0001 : 4] [Character Count Indicator : variable] [Data Bit Stream : 10*charcount]
0010	Alphanumeric	[0010 : 4] [Character Count Indicator : variable] [Data Bit Stream : 11*charcount]
0100	Byte encoding	[0100 : 4] [Character Count Indicator : variable] [Data Bit Stream : 8*charcount]
1000	Kanji encoding	[1000 : 4] [Character Count Indicator : variable] [Data Bit Stream : 13*charcount]
0011	Structured append	[0011 : 4] [Symbol Position : 4] [Total Symbols: 4] [Parity : 8]
0111	ECI	[0111 : 4] [ECI Assignment number : variable]
0101	FNC1 in first position	[0101 : 4] [Numeric/Alphanumeric/Byte/Kanji payload : variable]
1001	FNC1 in second position	[1001 : 4] [Application Indicator : 8] [Numeric/Alphanumeric/Byte/Kanji payload : variable]
0000	End of message	[0000 : 4]

Note:

Character Count Indicator depends on how many modules are in a QR code (Symbol Version).

ECI Assignment number Size:

8*1 bits if ECI Assignement Bitstream starts with '0'
8*2 bits if ECI Assignement Bitstream starts with '10'
8*3 bits if ECI Assignement Bitstream starts with '110'

5. Encoding modes can be mixed as needed within a QR symbol (for example, a url with a long string of alphanumeric characters).

[Mode Indicator][Mode bitstream] --> [Mode Indicator][Mode bitstream] --> etc... --> [0000 End of message (Terminator)]

6. After every indicator that selects an encoding mode is a length field that tells how many characters are encoded in that mode. The number of bits in the length field depends on the encoding and the symbol version.

Number of bits in a length field (Character Count Indicator)

Encoding	**Ver. 1–9**	**Ver. 10–26**	**Ver. 27–40**
Numeric	10	12	14
Alphanumeric	9	11	13
Byte	8	16	16
Kanji	8	10	12

7. Alphanumeric encoding mode stores a message more compactly than the byte mode can, but cannot store lower-case letters and has only a limited selection of punctuation marks, which are sufficient for rudimentary web addresses. Two characters are coded in an 11-bit value by this formula:

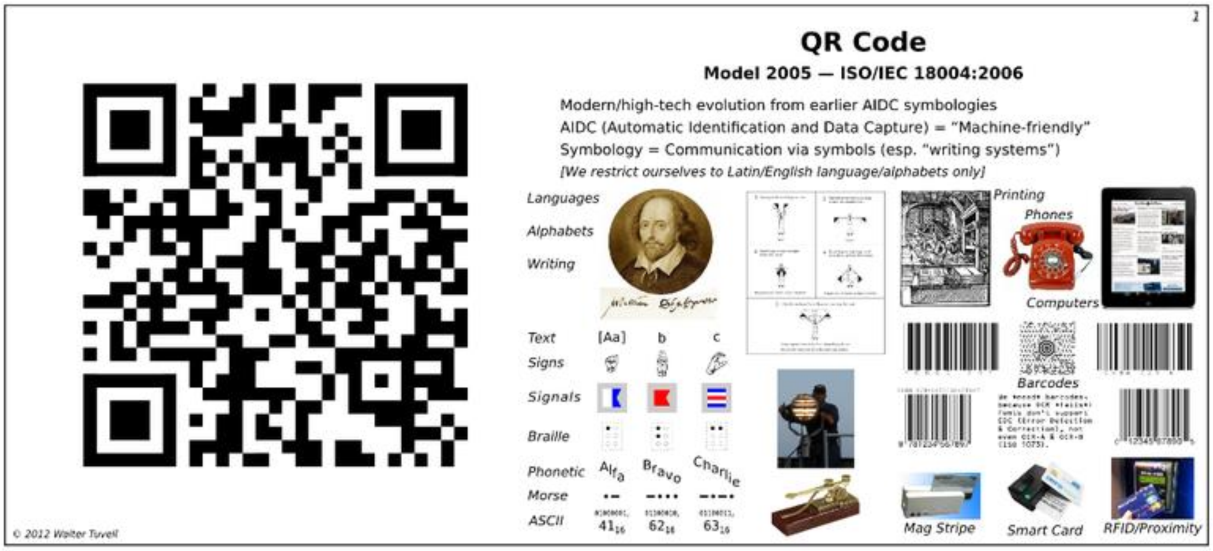
$$V = 45\,C_1 + C_2$$

Alphanumeric character codes

Code	Character	Code	Character	Code	Character	Code	Character	Code	Character
00	0	10	A	20	K	30	U	40	+
01	1	11	B	21	L	31	V	41	–
02	2	12	C	22	M	32	W	42	.
03	3	13	D	23	N	33	X	43	/
04	4	14	E	24	O	34	Y	44	:
05	5	15	F	25	P	35	Z		
06	6	16	G	26	Q	36	Space		
07	7	17	H	27	R	37	$		
08	8	18	I	28	S	38	%		
09	9	19	J	29	T	39	*		

Encoding:

The following boxes offer info on QR code decoding:



1

QR Code

Model 2005 — ISO/IEC 18004:2006

Modern/high-tech evolution from earlier AIDC symbologies

AIDC (Automatic Identification and Data Capture) = "Machine-friendly"

Symbology = Communication via symbols (esp. "writing systems")

[We restrict ourselves to Latin/English language/alphabets only]

Languages
Alphabets
Writing
Text
[Aa]
b
c
Signs
Signals
Braille
Phonetic
Alfa
Bravo
Charlie
Morse
ASCII
41₁₆
62₁₆
63₁₆
Printing
Phones
Computers
Barcodes
Mag Stripe
Smart Card
RFID/Proximity

Additional Notes

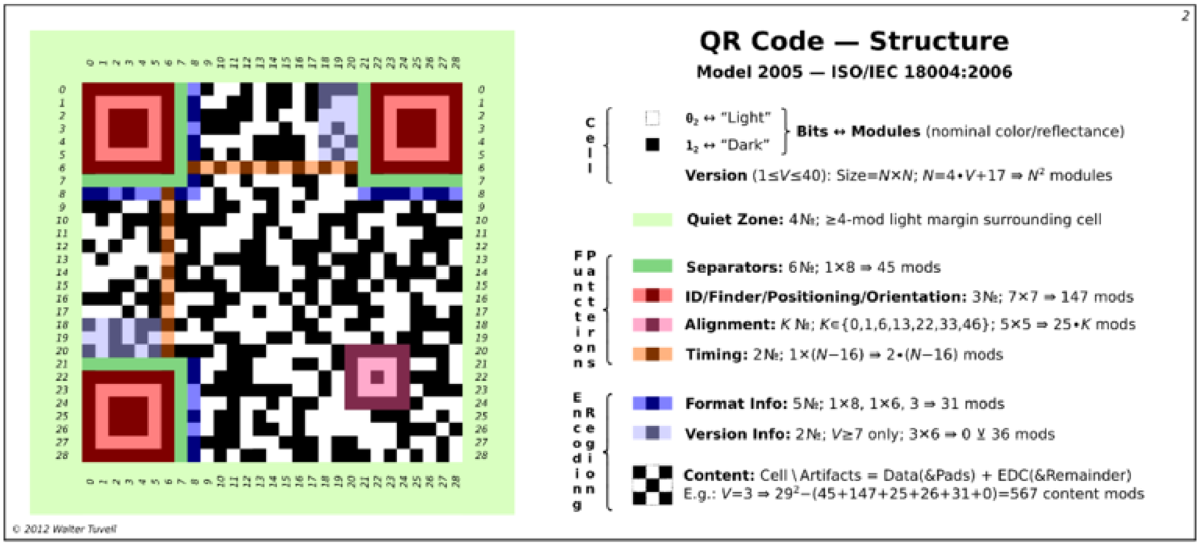
Invented in 1994 by Japanese company Denso-Wave (major funding Denso corp., minor funding Toyota).

Quickly entered standardization process, becoming ISO/IEC international standard in 2000. Current version is QR Code Model 2005.

Royalty-free usage, worldwide.

For commercial use, hardware scanners available in several technologies; no more expensive than other barcode scanners.

For casual use, software generators and readers are cheaply/freely available; apps for PCs and cell phones (working with images or cameras), and even web resources.



2

QR Code — Structure

Model 2005 — ISO/IEC 18004:2006

□ 0_2 ↔ "Light"

■ 1_2 ↔ "Dark"

Bits ↔ Modules (nominal color/reflectance)

Version $(1 \le V \le 40)$: Size$=N \times N$; $N=4 \cdot V+17 \Rightarrow N^2$ modules

Quiet Zone: 4№; ≥4-mod light margin surrounding cell

Separators: 6№; $1 \times 8 \Rightarrow 45$ mods

ID/Finder/Positioning/Orientation: 3№; $7 \times 7 \Rightarrow 147$ mods

Alignment: K №; $K \in \{0,1,6,13,22,33,46\}$; $5 \times 5 \Rightarrow 25 \cdot K$ mods

Timing: 2№; $1 \times (N-16) \Rightarrow 2 \cdot (N-16)$ mods

Format Info: 5№; 1×8, 1×6, $3 \Rightarrow 31$ mods

Version Info: 2№; $V \ge 7$ only; $3 \times 6 \Rightarrow 0 \veebar 36$ mods

Content: Cell \ Artifacts = Data(&Pads) + EDC(&Remainder)
E.g.: $V=3 \Rightarrow 29^2-(45+147+25+26+31+0)=567$ content mods

Additional Notes

"Light"/"dark" are understood relative to per-symbol global threshold (reflectance midway between minimum and maximum for the symbol). Polarity can be reversed (0_2↔"dark", 1_2↔"light") — if decoder cannot locate finder patterns, try reversing polarity (only one polarity can be valid). Other variations may also be supported, depending on the reader/decoder (e.g., physical/3D markings, such as dice-like dot/peen dimples). Easily depicted in ASCII, too (using *, ␣).

Corner features (quiet zone, finder patterns, separators) enable identification, location, rotational orientation, reflectance polarity.

Timing patterns (alternating reflectance) supports symbol density, version/size to be determined, and provide positions for determining row/col coordinates (rectangular, not just square).

Alignment patterns, located symmetrically about diagonal, supports symbol distortions to be detected/corrected.
№: 0 $(V=1)$; 1 $(2 \le V \le 6)$; 6 $(7 \le V \le 13)$; 13 $(14 \le V \le 20)$; 22 $(21 \le V \le 27)$; 33 $(28 \le V \le 34)$; 46 $(35 \le V \le 40)$.

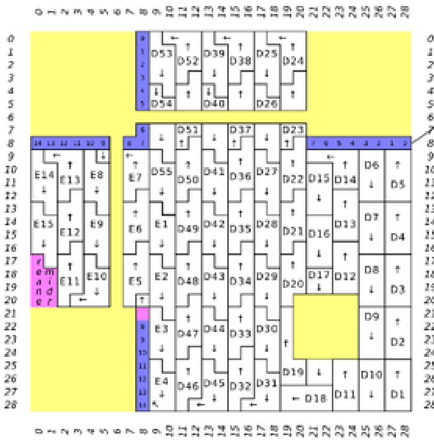
Version info: $7 \le V \le 40$ only (not present in our sample); 2 copies, each 18 bits (6 data, 12 EDC); encodes value V ; special-snake/zigzag layout; unmasked.

Version info EDC uses binary Golay 18,6 ECC (≤3-bit correction). The $\mathbb{F}_2[X]$ polynomial of degree ≤ 6 whose coefficients are bits 17–12 (which encode V), multiplied by $X^{18-6}=X^{12}$, is divided by designated generator polynomial $X^{12}+X^{11}+X^{10}+X^9+X^8+X^5+X^2+1$, and the remainder polynomial's coefficients are taken as bits 11–0. See ISO/IEC spec, §D.2, for error detection/correction procedure.

Damage to fixed patterns (deviation from ideal appearance of corner features, timing patterns, alignment patterns) can be evaluated, for quality control purposes (see ISO/IEC 15415 Print Quality specification).

QR Code — Layout & Stream-Encoding

Model 2005 — ISO/IEC 18004:2006



D53 D52 D39 D38 D25 D24
D54 D40 D26
D51 D37 D23
E14 E13 E8 E7 D55 D50 D41 D36 D27 D22 D15 D14 D6 D5
E15 E12 E9 E6 E1 D49 D42 D35 D28 D21 D16 D13 D7 D4
E11 E10 E5 E2 D48 D43 D34 D29 D20 D17 D12 D8 D3
r e m a i n d e r
E3 D47 D44 D33 D30 D9 D2
E4 D46 D45 D32 D31 D19 D18 D11 D10 D1

Format Info: 2№; 15 bits ⇒ 30 bits (+ 1 unused)

Bits 14–10 are *significant*; bits 9–0 are *derived* (EDC); *masked/"cooked"*

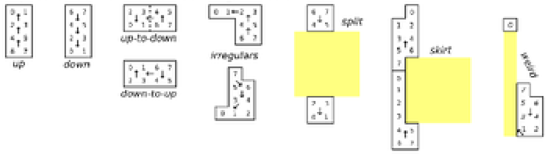
- **Level** (for EDC)
- **Mask** (for readability-robustness)

1 **unused** module; filled with 1_2 bit

Content ↠ D/E-Codespaces — Stream-Encoding Bit-Placement Principles:

- **Encoding Region** *tiled* by **codewords**; 8 bits; ~2-wide columns; directed (arrows)
- **Codeword-tiling** *snakes/zigzags bottom/right-to-top/left*, avoiding barriers; block-interleaved (enhances EDC; see ISO/IEC spec, §6.5.5–6, Table 9)
- **D/Data-space:** *Data* (prepared per protocol); streamed; *masked/"cooked"*
- **E/EDC-space:** *EDC; derived* from D-space; streamed; *remainder* modules (if any) filled with 0_2 bits; *masked/"cooked"*
- **Boundary** between D/E-spaces is determined by the EDC **level** in force
- **Bit-streaming:** MSb-to-LSb/right-to-left, in 2-wide directed order (arrows)
- **Bit-ordering:** $m \leq n$ ⇎ bit#$m \preccurlyeq$bit#n ("≤/less-than" ⇎ "≼/less-significant")

Examples of content bit-streaming ↠ codewords:



Additional Notes

The unused module is marked dark (and not masked).

"Snake" refers to byte-level layout; "zigzag" refers to bit-level layout.

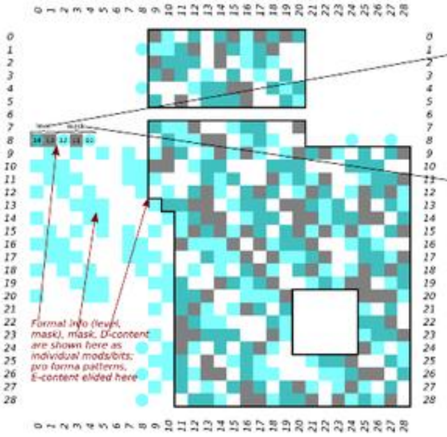
Layout/boundaries of D/E codewords is not to be confused with layout/boundaries of encoded charset SDD entities (see below).

Format info EDC uses BCH (Bose-Chaudhuri-Hocquenghem) 15,5 ECC (≤3-bit correction). The $\mathbb{F}_2[X]$ polynomial of degree ≤ 5 whose coefficients are bits 14–10 (level, mask), multiplied by $X^{15-5}=X^{10}$, is divided by designated generator polyno mial $X^{10}+X^8+X^5+X^4+X^2+X+1$, and the remainder polynomial's coefficients are taken as bits 9–0. See ISO/IEC spec, §C.2, for error detection/correction procedure.

Upon decoding, if format info doesn't validate, try mirror-reversal of symbol (= transpose about diagonal, exchanging rows and columns — which reverses the format info bits provided the value of bit 7 is 1_2, but forces bit 7 to be regarded as an error if bit 7 is 0_2). Both mirror images cannot be valid, to within 2 errors, due to design of format info pattern (by inspection of ISO/IEC spec Table C.1, no reversal of a valid format info string is also a valid string, to within 2 errors).

QR Code — Levels & Masks

Model 2005 — ISO/IEC 18004:2006



Formal info (level, mask), mask, D-content are shown here as individual mods/bits; pro forma patterns, E-content elided here

EDC Level: Determines count/distribution of codewords in D-space & E-space

Levels: 01_2=1≙**L**[ow]; 00_2=0≙**M**[edium]; 11_2=3≙**Q**[uality]; 10_2=2≙**H**[igh]

Detailed version/level definitions at ISO/IEC spec, §6.4.10, Table 7

E.g.: Version=3 ⇒ 567 content mods/bits ⇒ 70 codewords (+ 7 remainder bits)

VersionLevel=3-L ⇒ 55 D-words + 15 E-words
VersionLevel=3-M ⇒ 44 D-words + 26 E-words
VersionLevel=3-Q ⇒ 34 D-words + 36 E-words
VersionLevel=3-H ⇒ 26 D-words + 44 E-words

Raw format info (level, mask) also masked ("cooked") — not by the D/E mask (below), but by this special 15-bit ***Format Mask:*** *1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 ₂*

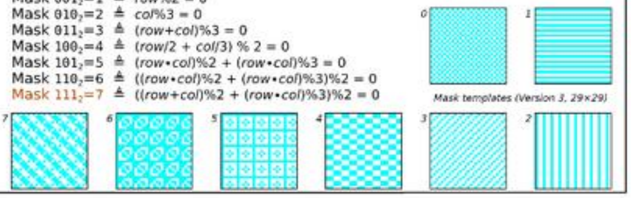
Mask: *Raw (not metadata)* D/E XOR-*masked ("cooked")* with **mask template**

Goal: Robustness for scanning/reading devices; balance light/dark modules; minimize occurrence of *pro forma* patterns in D/E spaces (*not* "crypto")

Robustness evaluation rubric defined at ISO/IEC spec, §6.8.1

D/E mask templates ('/' ≙ integer divison; '%' ≙ integer remainder):

Mask 000_2=0 ≙ *(row+col)*%2 = 0
Mask 001_2=1 ≙ *row*%2 = 0
Mask 010_2=2 ≙ *col*%3 = 0
Mask 011_2=3 ≙ *(row+col)*%3 = 0
Mask 100_2=4 ≙ *(row/2 + col/3)* % 2 = 0
Mask 101_2=5 ≙ *(row•col)*%2 + *(row•col)*%3 = 0
Mask 110_2=6 ≙ (*(row•col)*%2 + *(row•col)*%3)%2 = 0
Mask 111_2=7 ≙ (*(row+col)*%2 + *(row•col)*%3)%2 = 0

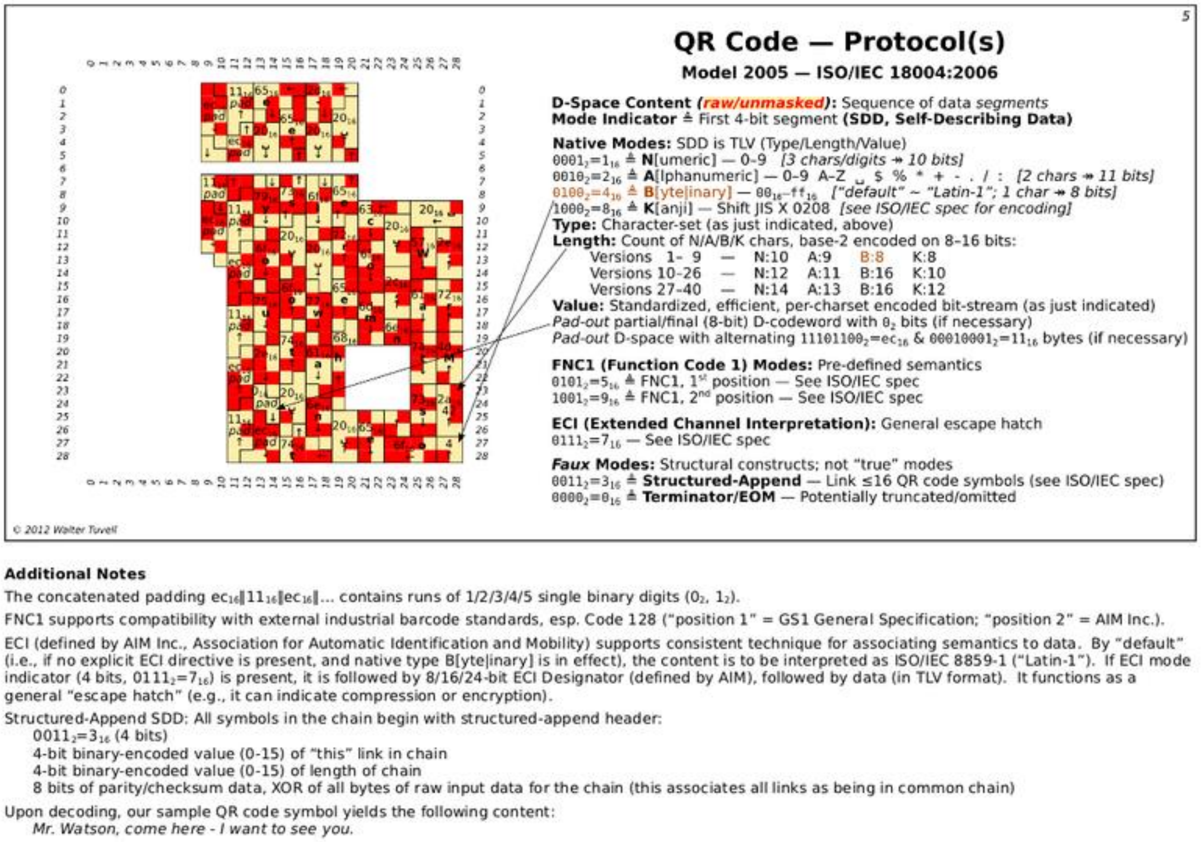


Additional Notes

The special mask 101,0100,0001,0010₂ ensures that masked/cooked format info is never all-0's.

D-space EDC uses RS (Reed-Solomon) *c,k,r* (for various values of *c*, *k*, *r*; see ISO/IEC spec, Table 9) ECC, over $\mathbb{F}_{256}=GF(2^8)$. The irreducible primitive polynomial defining $\mathbb{F}_{256}$ itself is taken to be $X^8+X^4+X^3+X^2+1$. 36 different generating polynomials are used (ISO/IEC spec, Table A.1). Error-correction capacity, *r* (number of correctable errors per block of codewords) varies, depending on versions 1–40 and levels L/M/Q/H, yielding error-correction percentages in ~7½% increments: approx. 7½/15/22½/30. See ISO/IEC spec, §6.5.2 and Annexes A–B for error-code generation and error detection/correction procedure.

$\mathbb{F}_{256}$-based scheme requires *blocking:* D-space must be initially partitioned into a number of blocks, of specified sizes (defined in ISO/IEC spec, Table 9); then each D-block is mapped to a corresponding E-block via ECC generation. Effect of blocking on placement of codewords is specified by ISO/IEC spec, §6.6 and Figure 15 (e.g., shortest block(s), if any, are placed first).



5

QR Code — Protocol(s)

Model 2005 — ISO/IEC 18004:2006

D-Space Content (*raw/unmasked*): Sequence of data *segments*
Mode Indicator $\triangleq$ First 4-bit segment **(SDD, Self-Describing Data)**

Native Modes: SDD is TLV (Type/Length/Value)
$0001_2=1_{16}$ $\triangleq$ **N**[umeric] — 0–9 *[3 chars/digits ↠ 10 bits]*
$0010_2=2_{16}$ $\triangleq$ **A**[lphanumeric] — 0–9 A–Z ␣ $ % * + - . / : *[2 chars ↠ 11 bits]*
$0100_2=4_{16}$ $\triangleq$ **B**[yte|inary] — 00_{16}–ff_{16} *["default" ~ "Latin-1"; 1 char ↠ 8 bits]*
$1000_2=8_{16}$ $\triangleq$ **K**[anji] — Shift JIS X 0208 *[see ISO/IEC spec for encoding]*
Type: Character-set (as just indicated, above)
Length: Count of N/A/B/K chars, base-2 encoded on 8–16 bits:

Versions 1- 9	—	N:10	A:9	B:8	K:8
Versions 10-26	—	N:12	A:11	B:16	K:10
Versions 27-40	—	N:14	A:13	B:16	K:12

Value: Standardized, efficient, per-charset encoded bit-stream (as just indicated)
Pad-out partial/final (8-bit) D-codeword with 0_2 bits (if necessary)
Pad-out D-space with alternating $11101100_2=ec_{16}$ & $00010001_2=11_{16}$ bytes (if necessary)

FNC1 (Function Code 1) Modes: Pre-defined semantics
$0101_2=5_{16}$ $\triangleq$ FNC1, 1st position — See ISO/IEC spec
$1001_2=9_{16}$ $\triangleq$ FNC1, 2nd position — See ISO/IEC spec

ECI (Extended Channel Interpretation): General escape hatch
$0111_2=7_{16}$ — See ISO/IEC spec

***Faux* Modes:** Structural constructs; not "true" modes
$0011_2=3_{16}$ $\triangleq$ **Structured-Append** — Link ≤16 QR code symbols (see ISO/IEC spec)
$0000_2=0_{16}$ $\triangleq$ **Terminator/EOM** — Potentially truncated/omitted

© 2012 Walter Tuvell

Additional Notes

The concatenated padding $ec_{16}\|11_{16}\|ec_{16}\|$… contains runs of 1/2/3/4/5 single binary digits (0_2, 1_2).

FNC1 supports compatibility with external industrial barcode standards, esp. Code 128 ("position 1" = GS1 General Specification; "position 2" = AIM Inc.).

ECI (defined by AIM Inc., Association for Automatic Identification and Mobility) supports consistent technique for associating semantics to data. By "default" (i.e., if no explicit ECI directive is present, and native type B[yte|inary] is in effect), the content is to be interpreted as ISO/IEC 8859-1 ("Latin-1"). If ECI mode indicator (4 bits, $0111_2=7_{16}$) is present, it is followed by 8/16/24-bit ECI Designator (defined by AIM), followed by data (in TLV format). It functions as a general "escape hatch" (e.g., it can indicate compression or encryption).

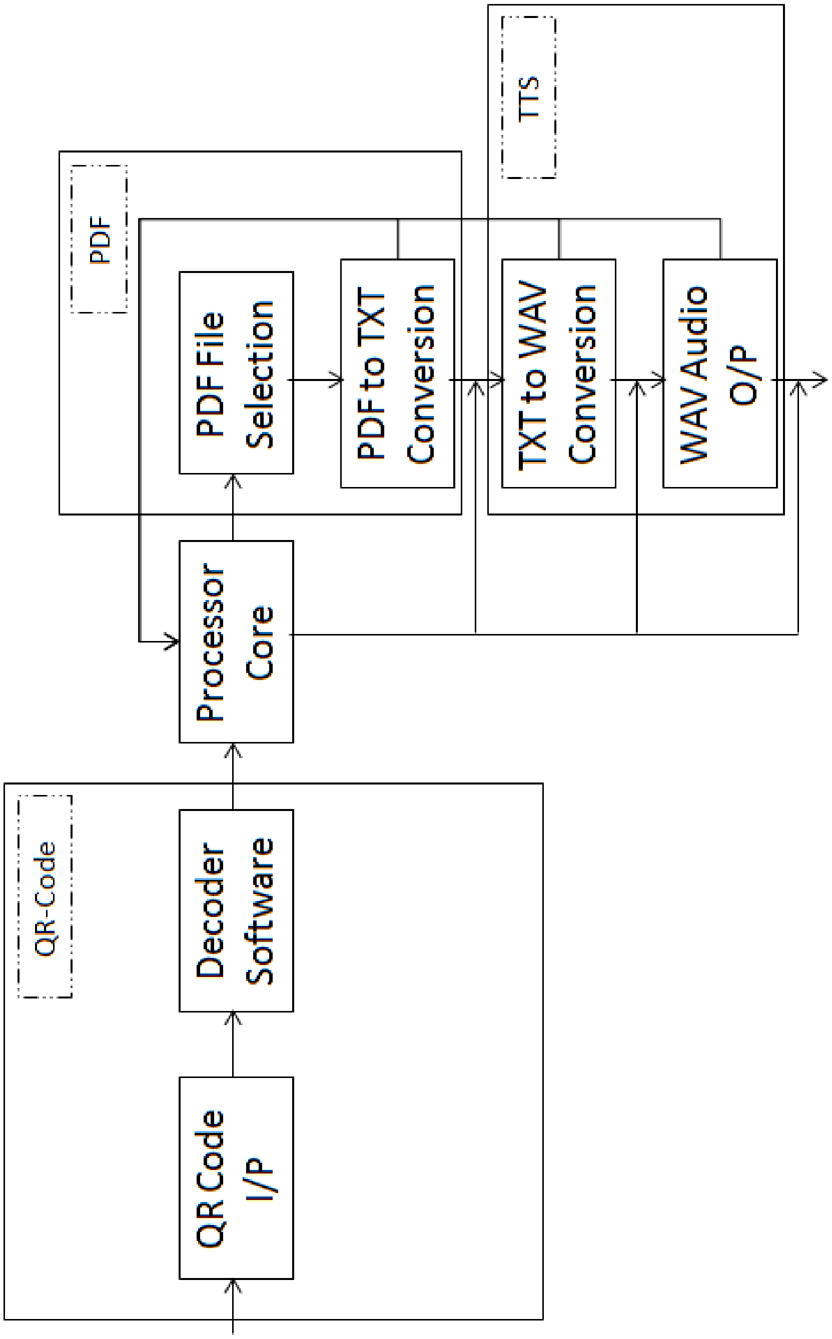
Structured-Append SDD: All symbols in the chain begin with structured-append header:
$0011_2=3_{16}$ (4 bits)
4-bit binary-encoded value (0-15) of "this" link in chain
4-bit binary-encoded value (0-15) of length of chain
8 bits of parity/checksum data, XOR of all bytes of raw input data for the chain (this associates all links as being in common chain)

Upon decoding, our sample QR code symbol yields the following content:
Mr. Watson, come here - I want to see you.

Risks:

1. The only context in which common QR codes can carry executable data is the URL data type. These URLs may host JavaScript code, which can be used to exploit vulnerabilities in applications on the host system, such as the reader, the web browser or the image viewer, since a reader will typically send the data to the application associated with the data type used by the QR code.
2. In the case of no software exploits, malicious QR codes combined with a permissive reader can still put a computer's contents and user's privacy at risk. This practice is known as "attagging", a portmanteau of "attack tagging". They are easily created and can be affixed over legitimate QR codes. On a smartphone, the reader's permissions may allow use of the camera, full Internet access, read/write contact data, GPS, read browser history, read/write local storage, and global system changes.
3. Risks include linking to dangerous web sites with browser exploits, enabling the microphone/camera/GPS, and then streaming those feeds to a remote server, analysis of sensitive data (passwords, files, contacts, transactions), and sending email/SMS/IM messages or DDOS packets as part of a botnet, corrupting privacy settings, stealing identity, and even containing malicious logic themselves such as JavaScript or a virus. These actions could occur in the background while the user is only seeing the reader opening a seemingly harmless web page

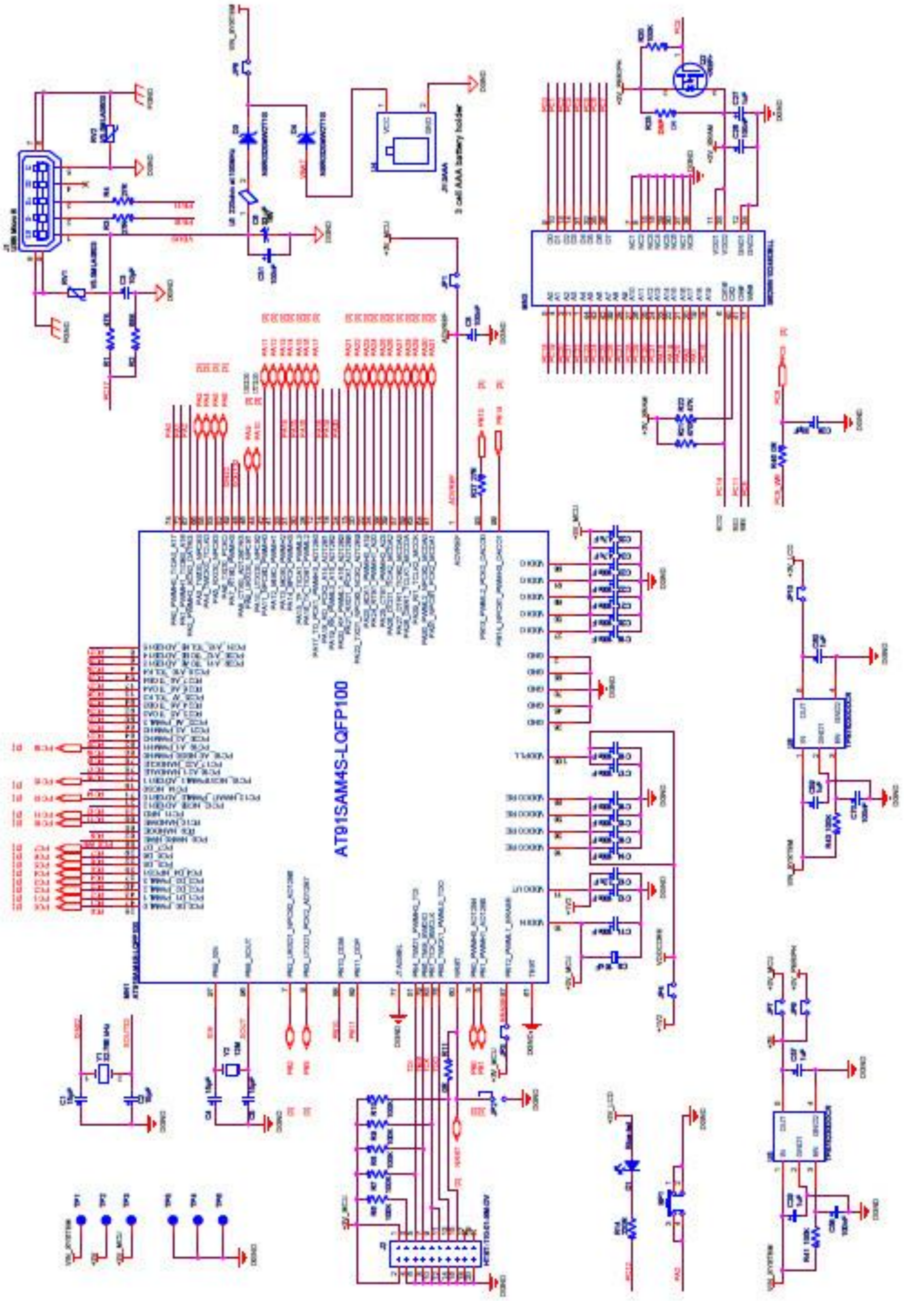
Block Diagram



PIA - Block Diagram

Fast Prototype

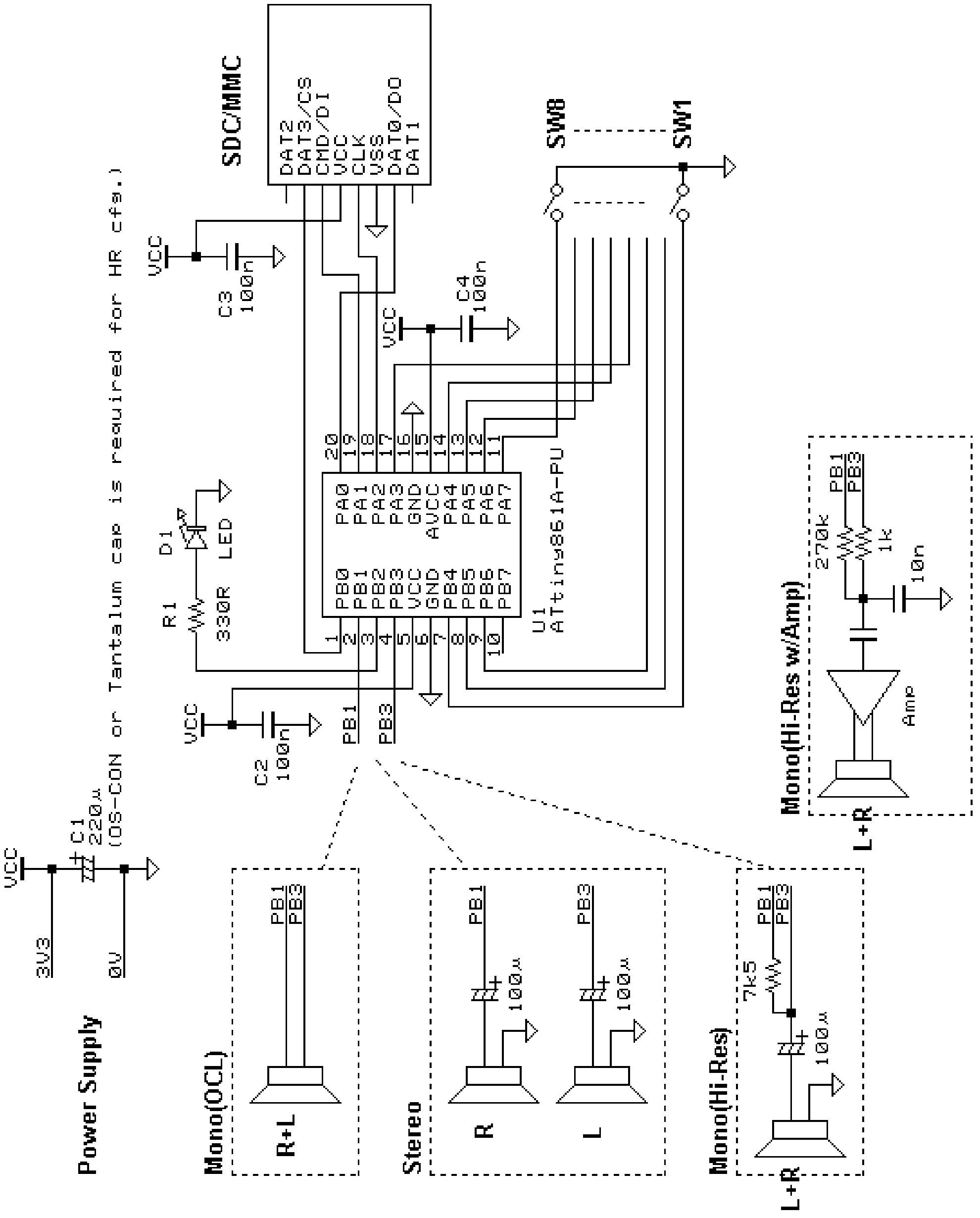
- **AT01180 Barcode & QR Code Scanner**
  (refer: http://www.atmel.com/tools/SAM4S-WPIR-RD.aspx?tab=documents)



Circuit Description:

This is a barcode & QR-code reader based on Atmel SAM4S_WPIR_RD kit. It carries out the processes of both scaning the image and decoding it. The decoded output could further be used to link to appropriate file(s) and transferring control to audio/display section.

- **PCM Sound Generator**
  (refer: http://elm-chan.org/works/sd20p/report)



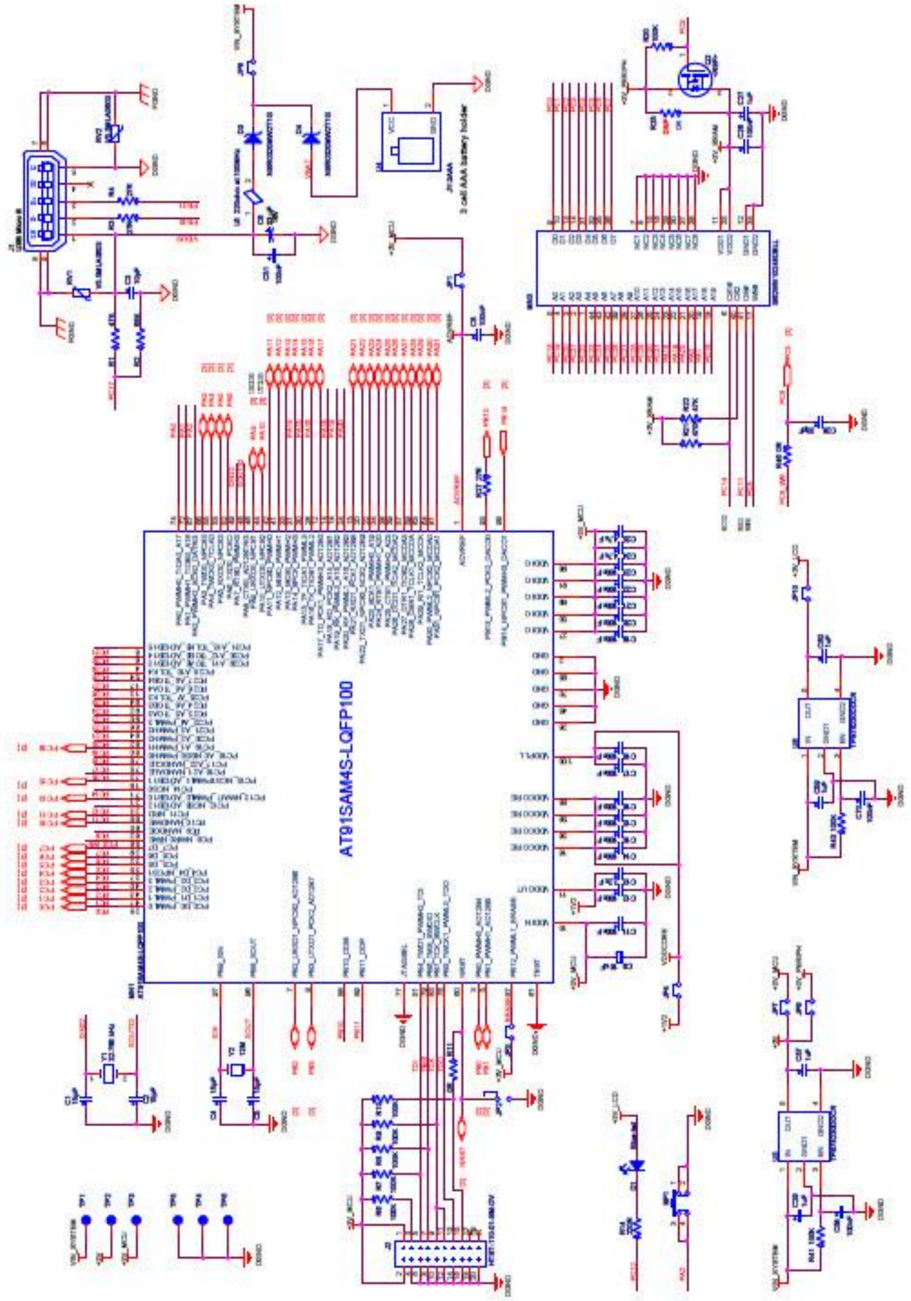
Circuit Description:

This is a simple sound generator based on 8-pin SD Audio Player. It is built with an Atmel ATtiny861 microcontroller and an SD memory card.

The Micro SD card is used as storage media with proper socket conversion. It is controlled in SPI mode via USI. The audio signal is output in PWM and it can drive loudspeakers directly. It may be thought that the output current exceeds the rating but it is no problem because the output current is limited by output impedance at 3.3 volt supply voltage even if the output pin is tied to the rail. Therefore it will create sufficient sound to be played on headphones.

Parts Selection

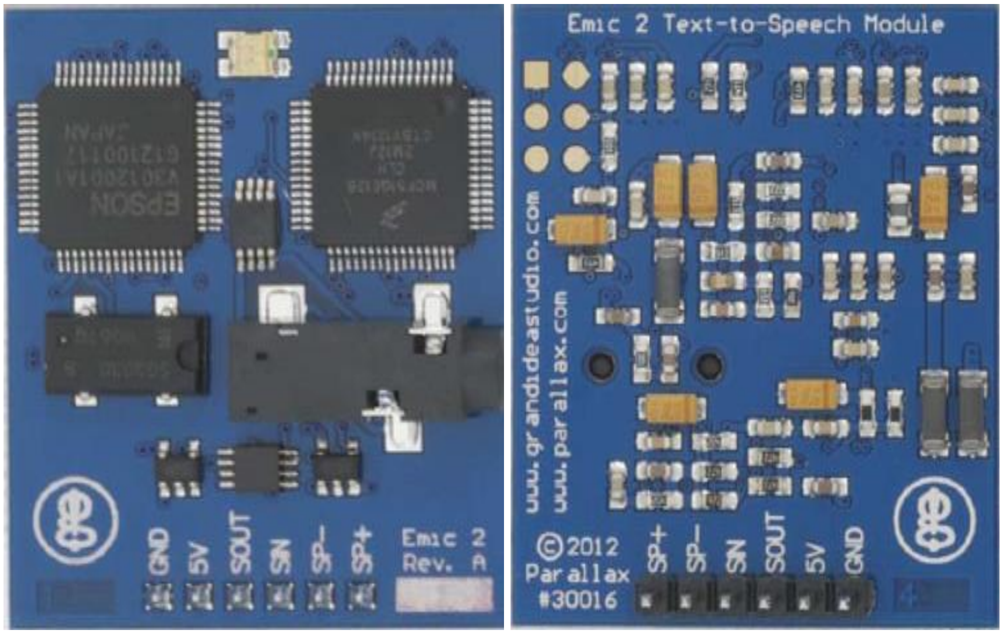
- **AT01180 Barcode & QR Code Scanner**
  (refer: http://www.atmel.com/tools/SAM4S-WPIR-RD.aspx?tab=documents)



Circuit Description:

This is a barcode & QR-code reader based on Atmel SAM4S_WPIR_RD kit. It carries out the processes of both scaning the image and decoding it. The decoded output could further be used to link to appropriate file(s) and transferring control to audio/display section.

- **Emic2 – TTS Module**
  (refer: https://www.parallax.com/product/30016)

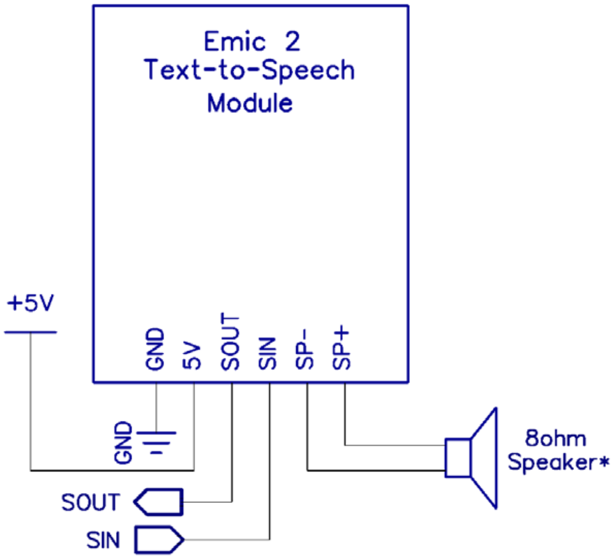


Description:

- The Emic 2 Text-to-Speech Module is an unconstrained, multi-language voice synthesizer that converts a stream of digital text into natural sounding speech output. Using the universally recognized DECtalk text-to-speech synthesizer engine, Emic 2 provides full speech synthesis capabilities for any embedded system via a simple command-based interface.

Usage:

- Emic 2 is controlled by the host via a serial communications interface. To use, simply send the desired command to Emic 2 and listen for audio output from the SP+/SP- speaker connection or 1/8" audio jack.



- The serial interface is configured for 9600 bps, 8 data bits, no parity, 1 stop bit (8N1). When Emic 2 is ready to receive commands, it will send a “:” to the host. It will then wait in an idle state until it receives a valid command, at which time it performs the command and returns any command-specific response. Emic 2 will return a “?” upon receiving an invalid command. Some microcontrollers with slow serial processing time, such as the BASIC Stamp 2, may miss the “:” response sent from the Emic 2 if the command completes quickly. This can be remedied by simply not checking for the response in the host program.
- On power-up, Emic 2 loads its default settings consisting of voice type, audio volume, speaking rate, language, and parser. These settings can be configured by the user to vary the audio output.

Progress Remarks

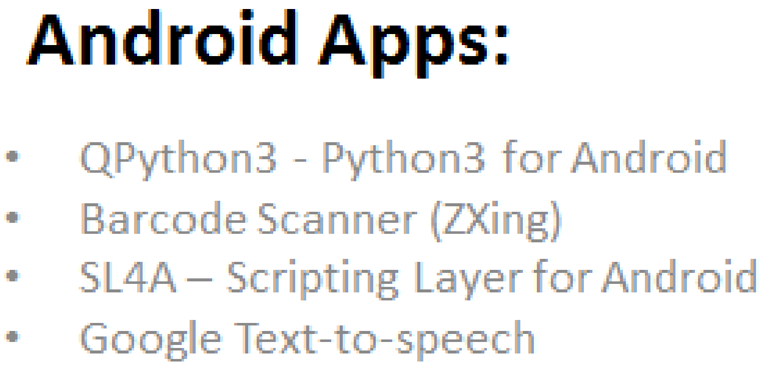
- **Unavailability of EMIC2 Module**
  Implementation of TTS module with the ATSAM4S main-board, or any other microcontroller was an easy solution to the problem of TTS conversion, but unavailability of the hardware made me turn up to choose an alternative in Raspberry Pi 3B.
  It is to be noted here that implementing a TTS program on a microcontroller is not very pretty since 8-bit data handling for PWM does not result in quality output. Also, an absence of operating system makes it difficult to implement file-management on the controller.
  RTOS could have been an alternative, but I went for RPi instead.

- **Hardware-independent Approach**
  Midway through the project, I came across a library for Android implementation – SL4A (Scripting Language For Android), which allows us to use android-device's resources along with Python (as a scripting language). So, I moved towards a hardware independent approach by implementing a solution on a python interpreter, using the Android framework.
  This project is thus, somewhat hardware-independent, and thus, almost free of cost for anyone having access to a modern Android device with a decent internet connectivity.

- **Source Code**
  The source-code will be made available (alongwith necessary datasets) on my Github page:
  http://www.github.com/CRT13/Projects/

Re-design

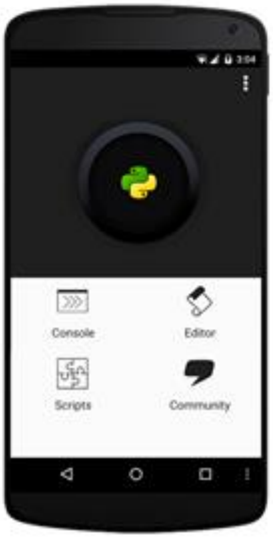
- **Hardware-independent Approach**
  Midway through the project, I came across a library for Android implementation – SL4A (Scripting Language For Android), which allows us to use android-device's resources along with Python (as a scripting language). So, I moved towards a hardware independent approach by implementing a solution on a python interpreter, using the Android framework.
  This project is thus, somewhat hardware-independent, and thus, almost free of cost for anyone having access to a modern Android device with a decent internet connectivity.



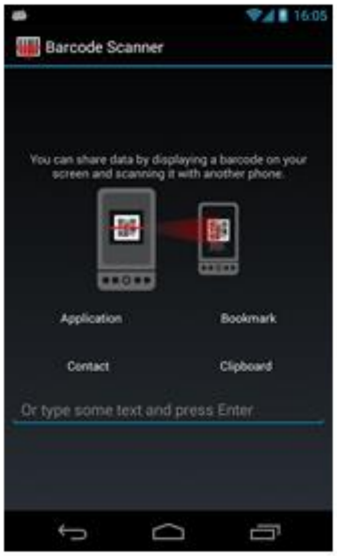
Android Apps:

- QPython3 - Python3 for Android
- Barcode Scanner (ZXing)
- SL4A – Scripting Layer for Android
- Google Text-to-speech

QPython3 - Python3 for Android

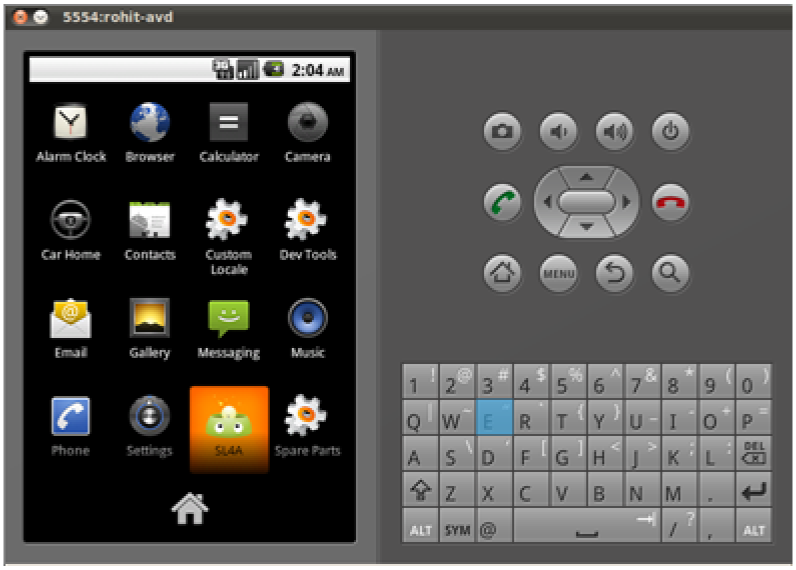


Barcode Scanner (ZXing)

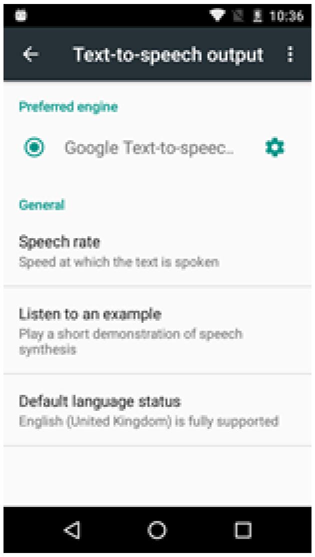


You can share data by displaying a barcode on your screen and scanning it with another phone.

SL4A – Scripting Layer for Android



Google Text-to-speech



Text-to-speech output

Preferred engine

Google Text-to-speec...

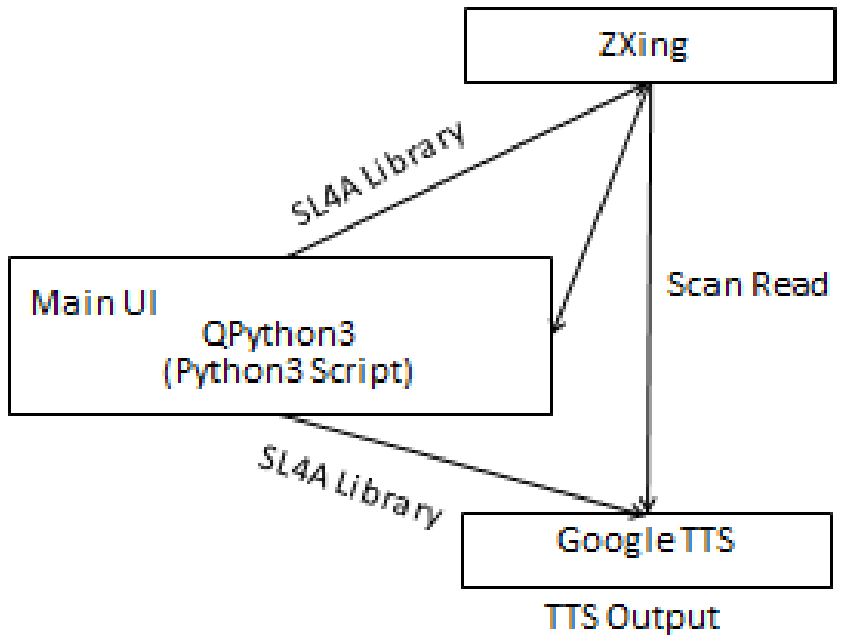
General

Speech rate
Speed at which the text is spoken

Listen to an example
Play a short demonstration of speech synthesis

Default language status
English (United Kingdom) is fully supported

App Flow


ZXing
SL4A Library
Main UI
QPython3
(Python3 Script)
Scan Read
SL4A Library
Google TTS
TTS Output

Example Programs

QRC2Speech:

1. Scan QR Code
2. Add QR Code to database
3. Remove QR Code to database

QRCMegaMart:

1. Add item to cart
2. Remove item from cart
3. Checkout

Source Code

- **QRC2Speech**

```
######################### IMPORT LIBRARIES #######################
import sl4a
import androidhelper
######################### INITIALIZATIONS #########################
droid1 = sl4a.Android()
droid2 = androidhelper.Android()
########################### SAMPLE DATA #########################
"""
qrcSMP = QRCode(Smartphones)
qrcCNT = QRCode(ClassNotes)
qrcYTB = QRCode(Youtube Links)
brcBOK = BarCode(Books)
"""
#################################################################
qrcSMP = ['SMPmice455','SMPgooPixXL','SMPonp3T','SMPhtc10','SMPsamGal6','SMPlenA7020a48','SMPlenK33A42','SMPyuYU5530','SMPlenA6020a46','SMPasuZC521TL']
qrcSMPweb = ['http://www.gsmarena.com/micromax_canvas_nitro_4g_e455-7665.php','http://www.gsmarena.com/google_pixel_xl-8345.php','http://www.gsmarena.com/oneplus_3t-8416.php','http://www.gsmarena.com/htc_10-7884.php','http://www.gsmarena.com/samsung_galaxy_s6-6849.php','http://www.gsmarena.com/lenovo_k5_note-7878.php','http://www.gsmarena.com/lenovo_k6_power-8317.php','http://www.gsmarena.com/yu_yunicorn-8002.php','http://www.gsmarena.com/lenovo_vibe_k5_plus-7947.php','http://www.gsmarena.com/asus_zenfone_3s_max_zc521tl-8550.php']

qrcYTB = ['YTBnptece01','YTBnptee01','YTBnptcse01','YTBnptme01','YTBnptce01','YTBnptase01','YTBnptche01','YTBnptphy01','YTBnptbio01','YTBnpthsc01']
qrcYTBweb = ['https://www.youtube.com/channel/UCdDLuSAR5CCBGDkgKuqPA3A','https://www.youtube.com/channel/UCTJn6buigC961hns17ELXAQ','https://www.youtube.com/channel/UCxJp9aEteKmOeobEsHXwxAw','https://www.youtube.com/channel/UCqBiwZuVP0NzSu4QR91v4jQ','https://www.youtube.com/channel/UCAVi5Zg6zSoyZUyKBtCJfmg','https://www.youtube.com/channel/UCiY8ERfD4qvD_-d-VUWh_GA','https://www.youtube.com/channel/UCqqc1GmsuANsx3s3Y0C-BsQ','https://www.youtube.com/channel/UC9vycSfjzLCR_w1C59VQo5A','https://www.youtube.com/channel/UCbWTmSK7bYM9kRZAdfy_gyg','https://www.youtube.com/channel/UCnLoshotJ_FRpzFmOChRPaQ']
#################################################################
########################### QRC Decode #########################
def qrcDecode():
   flag = True
```

```
txt = '.txt'
qrcValue = '/storage/emulated/0/qpython/CRT13/notes/'
print('==========================================')
print('1. Scan QR Code')
print('2. Add QR Code Entry')
print('3. Delete QR Code Entry')
print('==========================================')
usrInput = int(input('Please Select: '))
while flag is True:
    if usrInput < 0 or usrInput > 3:
        print('Error! Please try again')
        flag = False
    else:
        scan = droid1.scanBarcode().result
        # Scan QRCode
        if usrInput == 1:
            flag1 = True
            for i in range(0,10):
                if scan['extras']['SCAN_RESULT'] == qrcSMP[i]:
                    qrcValue += str(qrcSMP[i])
                    qrcValue += txt
                    file = open(qrcValue,'r')
                    for j in file:
                        droid1.ttsSpeak(j)
                    file.close()
                    droid2.view(str(qrcSMPweb[i]))
                    flag1 = False
                    break
                elif scan['extras']['SCAN_RESULT'] == qrcYTB[i]:
                    qrcValue += str(qrcYTB[i])
                    qrcValue += txt
                    file = open(qrcValue,'r')
                    for j in file:
                        droid1.ttsSpeak(j)
                    file.close()
                    droid2.view(str(qrcYTBweb[i]))
                    flag1 = False
                    break
                else:
                    pass
            if flag1:
                droid1.makeToast('No such entry exists!')
                droid1.ttsSpeak('Sorry, No such entry exists.Try making a new entry!')
                    break
            flag = False
        # Add QRCode
        elif usrInput == 2:
            flag1 = True
    qrcStr = str(scan['extras']['SCAN_RESULT'])
    if qrcStr[0:3] == 'SMP':
        for i in range(0,10):
            if qrcStr == str(qrcSMP[i]):
                droid1.makeToast('This entry already exists!')
                droid1.ttsSpeak('This entry already exists!')
                flag1 = False
                break
            else:
                pass
        if flag1:
            qrcSMP.append(qrcStr)
            print(qrcSMP)
            tmp = int(input('Add info about this QR code?(1/0)'))
            if tmp:
                qrcNewEntry(qrcStr)
            else:
                pass
            droid1.makeToast('Done!')
            droid1.ttsSpeak('Entry Made!')
            break
        break
    elif qrcStr[0:3] == 'YTB':
        for i in range(0,10):
            if qrcStr == str(qrcYTB[i]):
                droid1.makeToast('This entry already exists!')
                droid1.ttsSpeak('This entry already exists!')
                flag1 = False
                break
            else:
                pass
        if flag1:
            qrcYTB.append(qrcStr)
            print(qrcYTB)
            tmp = int(input('Add info about this QR code?(1/0)'))
            if tmp:
                qrcNewEntry(qrcStr)
            else:
                pass
            droid1.makeToast('Done!')
            droid1.ttsSpeak('Entry Made!')
            break
        break
    flag = False
# Delete QRCode
elif usrInput == 3:
    flag1 = True
    qrcStr = str(scan['extras']['SCAN_RESULT'])
```

```
                if qrcStr[0:3] == 'SMP':
                    for i in range(0,10):
                        if qrcStr == str(qrcSMP[i]):
                            del qrcSMP[i]
                            droid1.makeToast('Done!')
                            droid1.ttsSpeak('Entry Deleted!')
                            flag1 = False
                            break
                        else:
                            pass
                    if flag1:
                        droid1.makeToast('Entry does not exist!')
                        droid1.ttsSpeak('Sorry, no such entry exists!')
                        break
                    break
                elif qrcStr[0:3] == 'YTB':
                    for i in range(0,10):
                        if qrcStr == str(qrcYTB[i]):
                            del qrcYTB[i]
                            droid1.makeToast('Done!')
                            droid1.ttsSpeak('Entry Deleted!')
                            flag1 = False
                            break
                        else:
                            pass
                    if flag1:
                        droid1.makeToast('Entry does not exist!')
                        droid1.ttsSpeak('Sorry, no such entry exists!')
                        break
                    break
                flag = False
            txt = ''
            qrcValue = ''
########################## QRC New Entry ########################
def qrcNewEntry(qrcStr):
    try:
        qrcStr += '.txt'
        file = open('/storage/emulated/0/qpython/CRT13/notes/'+qrcStr,'w')
        usrInput = input('Enter Description:')
        file.write(usrInput)
        file.close()
    except:
        print('Error!')
#################################################################
########################## MAIN PROGRAM ########################
try:
    check = True
    count = 1
    print('=========================================')
    print('========= Personal Info Assistant ========')
    print('=========================================')
    while check:
        qrcDecode()
        check = int(input('Continue? (1/0):'))
except:
    print('Error!')
###############################################################
```

- **QRCMegaMart**

```
######################## IMPORT LIBRARIES #######################
import sl4a
import androidhelper
######################## INITIALIZATIONS ########################
droid1 = sl4a.Android()
droid2 = androidhelper.Android()
txt = '.txt'
########################### SAMPLE DATA #########################
"""
qrcITEMS = QRCode(All Available Items)
qrcCART = QRCode(Cart Items)
"""
#################################################################
qrcITEMS = ['ITEM00','ITEM01','ITEM02','ITEM03','ITEM04']
qrcCART = []
billITEMS = [60,70,100,50,300]
billCART = []
#################################################################
##################### FUNCTION DEFINITION #######################
def qrcUser(countMain):
  count = countMain
  countMain = QRCUser(count)
  if countMain != -1:
    print('You have',countMain,'items in your cart')
  return countMain
def QRCUser(count):
  flag = True
  bill = 0
  qrcValue = '/storage/emulated/0/qpython/CRT13/notes/'
  print('========================================')
  print('=========== Welcome to MegaMart ==========')
  print('========================================')
  print('1. Add To Your Cart')
  print('2. Remove From Your Cart')
  print('3. Checkout')
  print('========================================')
  usrInput = int(input('Please Select: '))
  while flag is True:
    if usrInput < 0 or usrInput > 3:
      print('Error! Please try again')
      flag = False
    elif usrInput == 3:
      print("You've Successfully Checked Out!")
      for i in range(0,count):
        bill += billCART[i]
      print('Your Total Bill is Rs.',bill)
```

```
         droid1.ttsSpeak('Your total bill is Rupees')
         droid1.ttsSpeak(str(bill))
         flag = False
         return -1
      else:
         scan = droid1.scanBarcode().result
         if usrInput == 1:
            for i in range(0,5):
               if scan['extras']['SCAN_RESULT'] == qrcITEMS[i]:
                  qrcCART.append(qrcITEMS[i])
                  billCART.append(billITEMS[i])
                  count +=1
                  qrcValue += str(qrcITEMS[i])
                  qrcValue += txt
                  file = open(qrcValue,'r')
                  droid1.ttsSpeak('You have successfully added')
                  for j in file:
                     droid1.ttsSpeak(j)
                  file.close()
                  qrcValue = ''
                  flag = False
         elif usrInput == 2:
                  if count == 0:
                     print('You have no items left!')
                     flag = False
                  for i in range(0,5):
                     if scan['extras']['SCAN_RESULT'] == qrcITEMS[i]:
                        for n in range(0,count):
                           if qrcCART[n] == qrcITEMS[i]:
                              del qrcCART[n]
                              del billCART[n]
                              count -= 1
                              qrcValue += str(qrcITEMS[i])
                              qrcValue += txt
                              file = open(qrcValue,'r')
                              droid1.ttsSpeak('You have successfully removed')
                              for j in file:
                                 droid1.ttsSpeak(j)
                              file.close()
                              qrcValue = ''
                              flag = False
                              break
   return count
#################################################################
```

```
########################## MAIN PROGRAM #########################
try:
    countMain = 0
    check = True
    while check is True:
        chkinput = int(input('Enter 0 to continue...'))
        if chkinput == 0:
            count = qrcUser(countMain)
            countMain = count
        else:
            check = False
except:
    print('Error!')
#################################################################
```

Future Plan

- **QR Code based Automation**

The aim is to automate repetitive work in a large organization by means of an android app that works with QR codes.

For this, the target organization would be our engineering college.

Some issues that would be targetted are:

1. Student attendance system
2. Audio-based information assistance for laboratory apparatus
3. Department-wise resources for students. (Timetables, subject syllabi (pdf), PDFs, assignments, prev. midsem + endsem papers,
4. Text alerts in case of cancellation/rescheduling of classes
5. Student-teacher feedback system
6. Ordering on-the-go from college canteen. (Saves money on Point-of-Sale machines; menu can be accessed online)