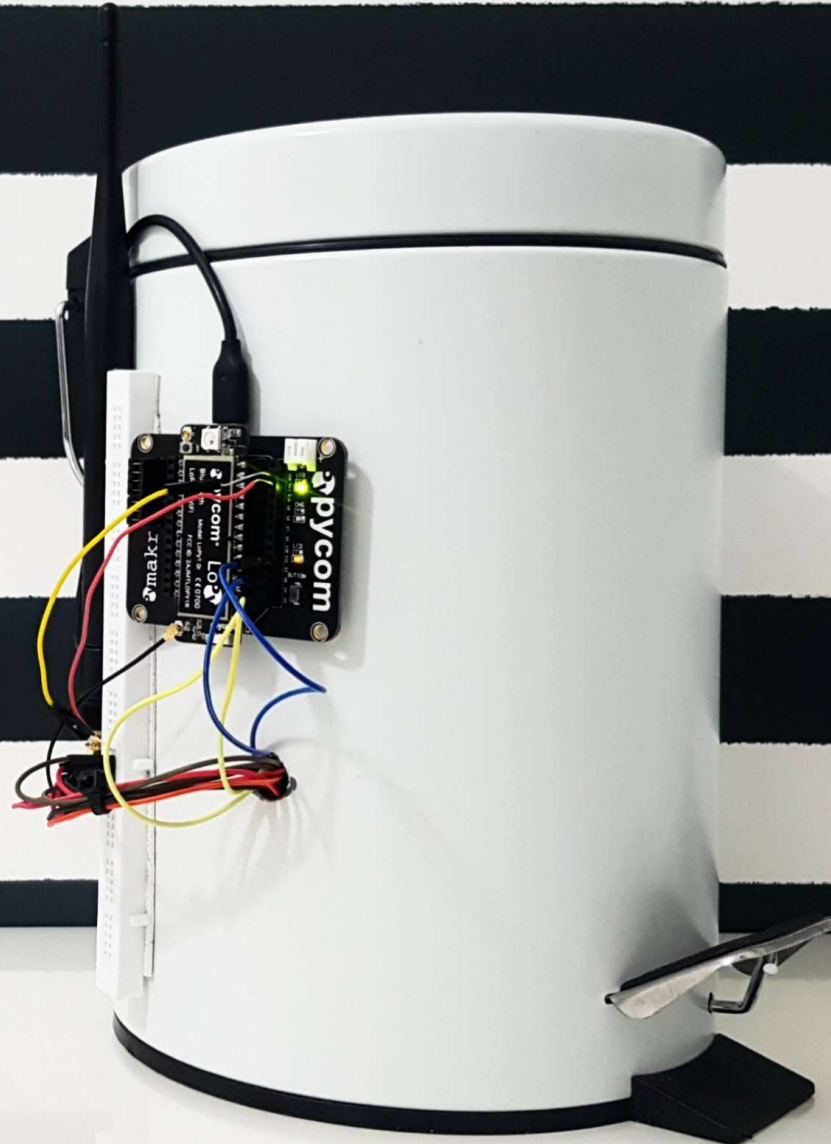


Can be civil



# Indice

- Introduzione
- Lopy client
- Raspberry server
- Bot Telegram
- Conclusione e sviluppi futuri

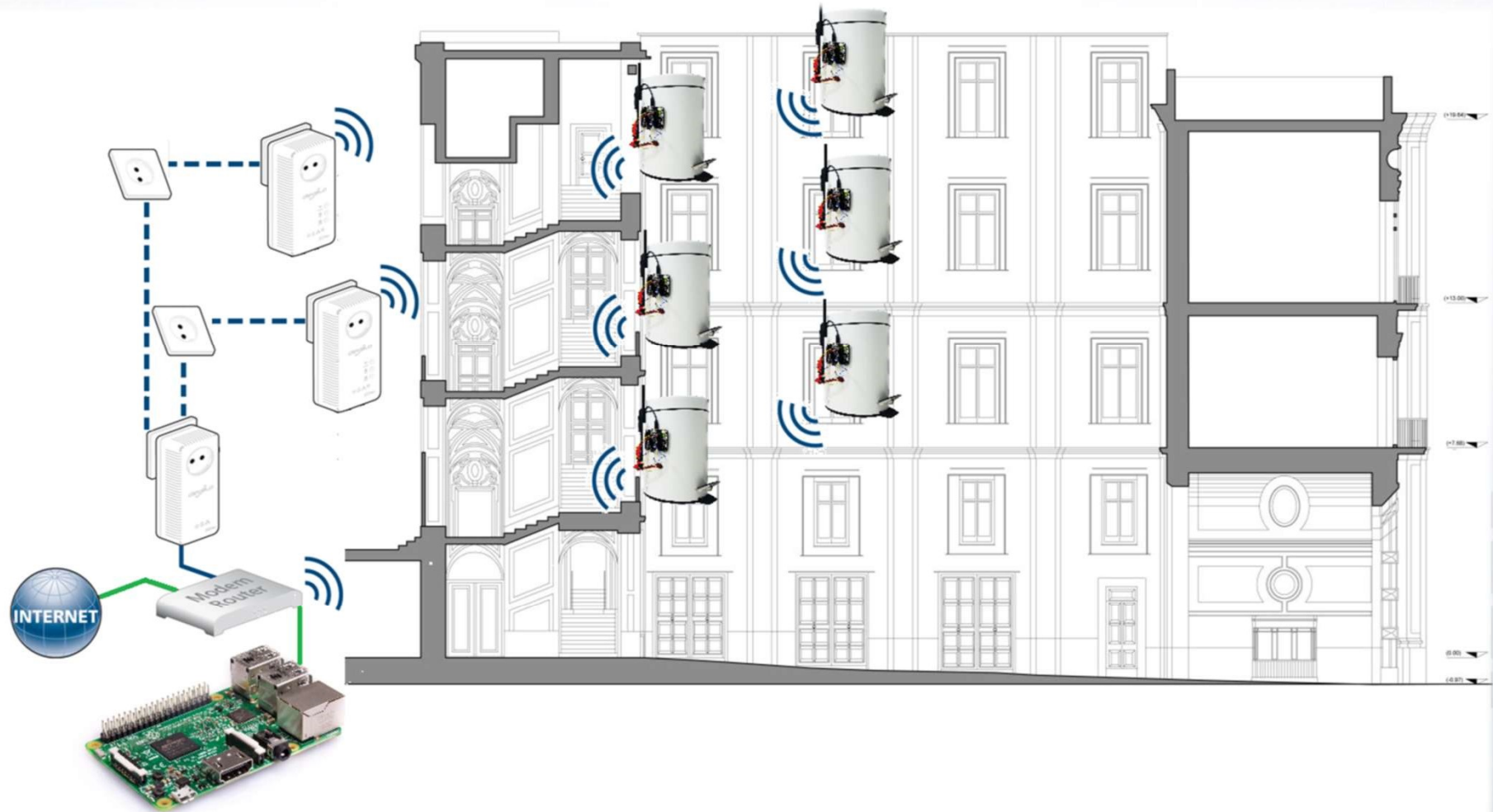
# Introduzione

L'attività di monitoraggio scelta per questo lavoro progettuale è basata sull'osservazione del livello di riempimento di contenitori atti alla raccolta dei rifiuti.

## **Funzionalità offerte:**

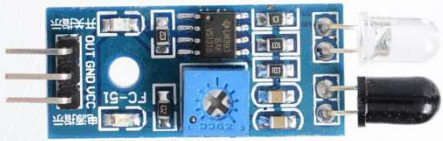
1. Notifica di avvertimento (BOT Telegram) quando il cestino risulta essere pieno almeno per metà
2. Notifica critica (BOT Telegram) per la segnalazione di cestino otturato. Di conseguenza, diviene necessario l'intervento umano di ripristino
3. Notifica critica (BOT Telegram) nella quale si richiede l'intervento di un operatore il cui compito è occuparsi dello svuotamento del cestino pieno

# Scenario





# Hardware necessario



**Sensori:** I sensori TE174 sfruttano la tecnologia IR per il rilevamento di oggetti in loro prossimità. Una coppia di led (Trasmettitore e Ricevitore) segnalano, in un ristretto campo d'azione, la presenza di oggetti nel caso in cui il segnale inviato dal trasmettitore venisse ostacolato e quindi riflesso sul ricevitore.



**LoPy:** Il dispositivo più vicino al fenomeno da osservare ed è l'unico in collegamento diretto con i sensori atti alla rilevazione di oggetti inseriti all'interno dei contenitori. L'obiettivo di questo componente è quello di elaborare i risultati ottenuti dai sensori ed effettuare segnalazioni opportune ed immediate. (Client)



**Raspberry:** Il computer progettato per ospitare sistemi operativi basati su kernel Linux; risulta molto utile al sistema perché capace di effettuare un collegamento con una rete di LoPy ad esso connessi. Riceve le segnalazioni di riempimento del cestino e gestisce le opportune notifiche da inoltrare. Per svolgere questo compito è necessario un server costruito Ad-hoc. (Server)

# Hardware necessario



**Router WiFi:** Permette la connessione tra Raspberry Pi e tutti i LoPy che dovranno comunicare con esso. Rende possibile, mediante una connessione internet, il servizio di notifica verso Telegram.



**Powerline:** Tecnologia per la trasmissione di dati che utilizza la rete di alimentazione elettrica come mezzo trasmissivo. Colma le problematiche relative alla copertura fornita dal segnale wireless del router.

# Analisi dei Costi

COMPONENTE	COSTO STIMATO (€)
Sensore di prossimità	2
Microcontrollore (Arduino/Lopy)	30
Computer (Raspberry)	30
Router Wifi	30
Powerline adapter (opzionale)	20

Dall'analisi dei costi di ogni singolo componente adoperato si evince come il sistema concepito, abbia un costo totale abbastanza limitato; questo dato sposa perfettamente quelle che sono le odierne strategie di marketing di numerose aziende o enti.

Tra i possibili scenari applicativi di questa tecnologia vi è certamente l'ambito aziendale o degli uffici in genere, nel quale, potrà essere notificato opportunamente il riempimento dei contenitori predisposti alla raccolta rifiuti, incrementando così il decoro e la pulizia complessiva.

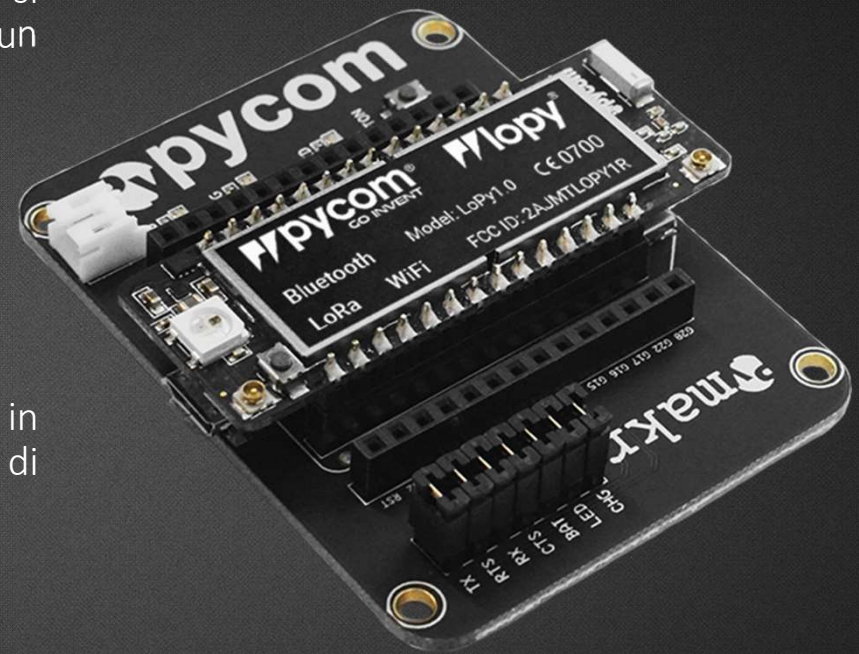


# LoPy Client



LoPy è una scheda di sviluppo WiFi, Bluetooth e LoRa con MicroPython abilitato, progettata specificamente per le applicazioni IoT. La scheda si basa sul sistema on-chip (SoC) Espressif ESP32 che presenta un microcontrollore dual core, WiFi, Bluetooth, LoRa e 512 KB di RAM.

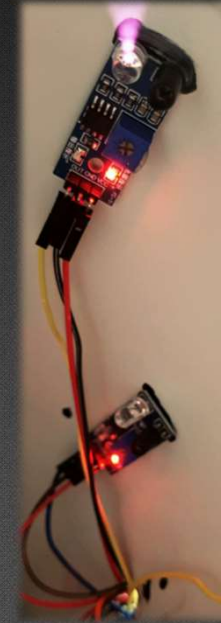
Nel nostro progetto, il LoPy è stato pensato per funzionare da Client in una rete locale sfruttando il protocollo 802.11, ai fini del controllo di due coppie di sensori di prossimità disposti all'interno del recipiente.





# LoPy Client

- I sensori di prossimità sono stati connessi al LoPy mediante l'utilizzo della Expansion Board :



- Assegnazione degli output dei sensori montati sul cestino alle variabili " s1, s2, s3, s4 " mediante l'utilizzo dei pin in dotazione alla Expansion Board

```
s1 = Pin('G5',mode=Pin.IN,pull=Pin.PULL_UP)
s2 = Pin('G4',mode=Pin.IN,pull=Pin.PULL_UP) #G4 e G5 Top Bin
s3 = Pin('G0',mode=Pin.IN,pull=Pin.PULL_UP)
s4 = Pin('G31',mode=Pin.IN,pull=Pin.PULL_UP) #G0 e G31 Middle Bin
```



# LoPy Client

- Istanziamento del socket che verrà utilizzato per comunicare con il Raspberry Pi che svolgerà il compito di server.

La costante `socket.AF_INET` serve a specificare il dominio sul quale il network funzionerà ovvero IPv4

La costante `socket.SOCK_STREAM` indica che stiamo creando una socket per una connessione TCP

```
clientsocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

- Connessione del client all'indirizzo IP locale del Raspberry Pi, sulla porta dedicata al server in ascolto.

```
clientsocket.connect(('192.168.43.114', 8089))
```

Il corpo del programma è principalmente composto da controlli sul contenitore che all'occorrenza attivano le seguenti funzionalità :

- Accensione di un led (Blu, Rosso o Giallo) che identifica lo stato del contenitore
- Inoltro di un messaggio mediante il socket

```
if (s3()==0 and s4()==0) :  
    print("Cestino mezzo pieno")  
    pycom.heartbeat(False)  
    pycom.rgbled(0x000099)  
    while bool1 == 0 :  
        clientsocket.send('Ces1')  
        bool1 = 1
```

# Raspberry Server



Il Raspberry Pi è un single-board computer a basso costo sviluppato dalla Raspberry Pi Foundation e originariamente pensato per stimolare l'interesse nell'informatica tra gli adolescenti. Oggi, esso non offre la potenza di un computer desktop, ma può essere utilizzato per tutte quelle attività che non necessitano delle elevate prestazioni di un PC costoso e ingombrante.

Nella nostra implementazione progettuale si è pensato di adoperare un Raspberry Pi 3 in quanto integra il WiFi 802.11n, protocollo di comunicazione utile alla connessione del LoPy che funge da client e dello stesso Raspberry il quale agisce da Server. Quest'ultimo, riceve le richieste di segnalazione dai client e offre un servizio che permette l'invio mediante un BOT Telegram dei messaggi di notifica.





# Raspberry Server



- Creazione di un'istanza di TCP/IP socket

```
print("\n Buongiorno, questo server monitora la situazione dei cestini.\n")
serversocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print 'Socket created'
```

- Bind dell'indirizzo IP e della porta (non well-known) e assegnazione all'istanza serversocket

```
try:
    serversocket.bind(('192.168.43.114', 8089))
except socket.error as msg:
    print 'Bind failed. Error Code : ' + str(msg[0]) + ' Message ' + msg[1]
    sys.exit()

print 'Socket bind complete'
print("\n Il server e' in attesa di una connessione \n")
```

- Abilitazione del server socket, che potrà accettare al più 10 connessioni

```
serversocket.listen(10)
```

- Viene accettata una connessione tra il server e il client

```
connection, address = serversocket.accept()
print ('\nConnected to ' + address[0] + ':' + str(address[1]) + '\n')
```

# Raspberry Server



Server in ascolto

```
while True:
    buf = connection.recv(64)
    if (buf=="Ces1"):
        print 'Cestino mezzo pieno'
        os.system('curl -s -X POST https://api.telegram.org/bot509765588:AAG76DK3MQP54_PCo5dCOpDvDM7ZHvFwmXk/sendMessage
        -d chat_id=-226231007 -d text="Attenzione Prego: Si registra cestino mezzo pieno" > /dev/null')

    elif(buf=="Ces2"):
        print 'Cestino pieno 100%'
        os.system('curl -s -X POST https://api.telegram.org/bot509765588:AAG76DK3MQP54_PCo5dCOpDvDM7ZHvFwmXk/sendMessage
        -d chat_id=-226231007 -d text="Cestino pieno al 100%. Provvedere al ricambio busta" > /dev/null')
        connection, address = serversocket.accept()
        print ('\nConnected to ' + address[0] + ':' + str(address[1]) + '\n')

    elif(buf=="Ces3"):
        print 'Cestino otturato'
        os.system('curl -s -X POST https://api.telegram.org/bot509765588:AAG76DK3MQP54_PCo5dCOpDvDM7ZHvFwmXk/sendMessage
        -d chat_id=-226231007 -d text="Cestino otturato. Provvedere al ripristino" > /dev/null')
        connection, address = serversocket.accept()
        print ('\nConnected to ' + address[0] + ':' + str(address[1]) + '\n')

serversocket.close()
```

Notifica di «Cestino mezzo pieno»

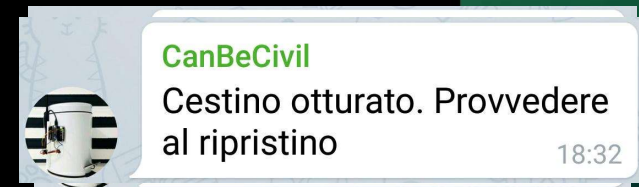
Notifica di «Cestino pieno»

Notifica di «Cestino otturato»

# Raspberry Server



```
pi@raspberrypi:~/Desktop $ python ServerOfficial.py
```





# Bot Telegram



Telegram è una piattaforma di *instant messaging* che offre la possibilità di creare dei BOT e di interagire con questi. Per creare il BOT è necessario aggiungere ai propri contatti il profilo **BotFather** che fungerà da guida durante il processo di creazione.

Sarà richiesto di inserire un nome per il nuovo BOT e successivamente uno username, che deve terminare con "bot".

Quando si riuscirà ad inserire uno username verrà fornito il *token* per le API basate su HTTP. Questo, non dovrà essere distribuito perché sarà la chiave di accesso per gestire e programmare il BOT. Il nuovo BOT è stato creato!

*@Canbecivilbot*



# Bot Telegram



Per inviare un messaggio attraverso le API di Telegram, il BOT necessita di conoscere l'ID della chat per cominciare una conversazione. L'ID sarà generato nel momento in cui ha inizio la prima comunicazione con il BOT.

Per ottenere l'ID della chat, è sufficiente recarsi al seguente URL:

<https://api.telegram.org/bot<TOKEN>/getUpdates>

```
"message":{"message_id":504,"from":
{"id":123456789,"is_bot":false,"first_name
":"John","last_name":"Doe","language_co
de":"it"},"chat":
{"id":-123456789,"title":"Can Be
Civil","type":"group","all_members_are_adm
inistrators":true},"date":1515182325,"text
":"/start","entities":
[{"offset":0,"length":6,"type":"bot_comman
d"}]}}
```

L'intuitività dell'API di Telegram rende semplice utilizzarla da script o app che eseguono attività automatizzate; l'invio automatico di un messaggio è reso possibile in maniera immediata grazie all'esecuzione del comando *curl* importato su ambienti Linux :

```
curl -s -X POST https://api.telegram.org/bot<TOKEN>/sendMessage -d chat_id= " " -d text="messaggio"
```

# Conclusione e Sviluppi Futuri

In *conclusione*, grazie a questo sistema, è possibile controllare dei comuni oggetti come i cestini della spazzatura che, essendo dotati di un sistema in grado di valutarne lo stato di riempimento, possono trasmettere informazioni rendendo ottimale la gestione dello svuotamento; sarà così evitato che qualche cestino trabocchi mentre altri vengano svuotati quando non è necessario.

Fra i *vantaggi* offerti da questa tipologia di sistema troviamo:

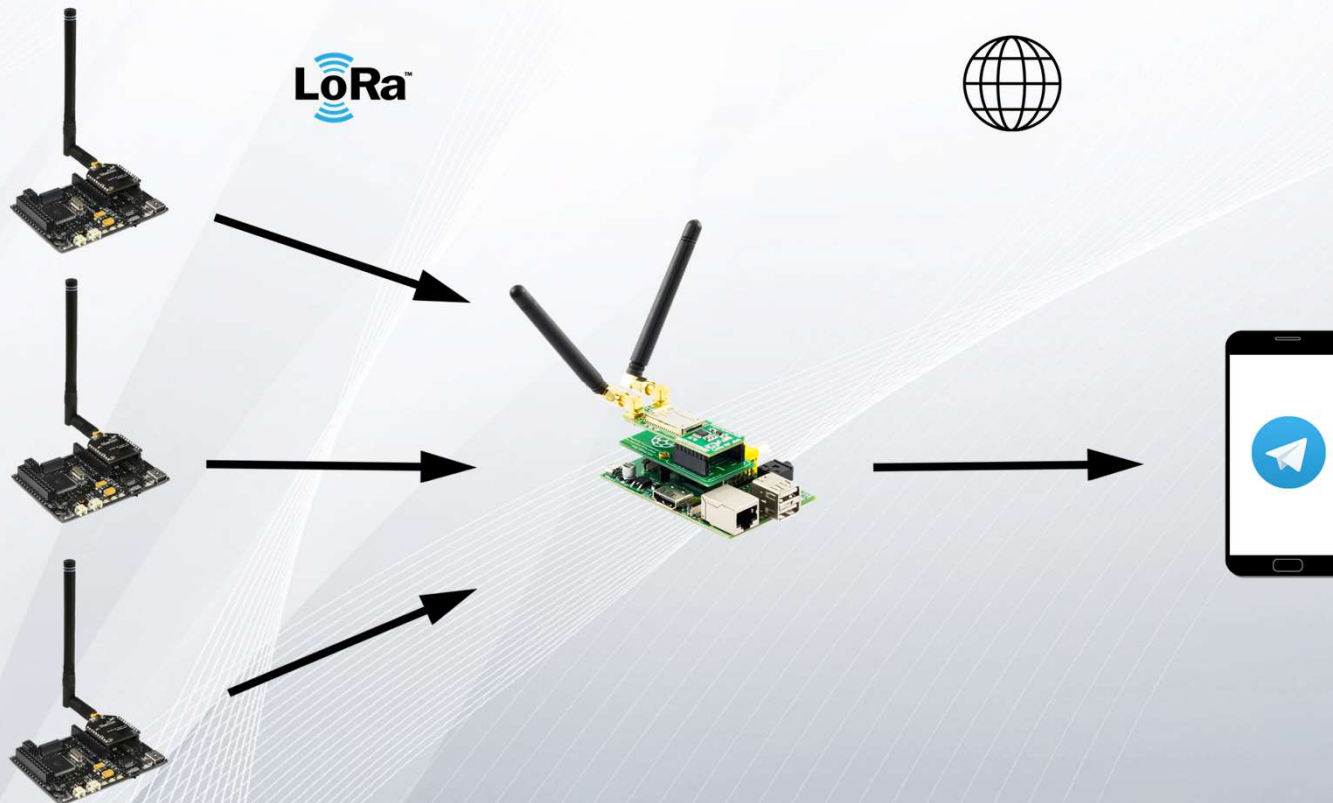
- Qualità degli ambienti: gli utilizzatori non trovano più dei recipienti colmi e pertanto inutilizzabili.
- Ottimizzazione della raccolta rifiuti: riduzione dei costi e riorganizzazione dinamica degli itinerari di raccolta




# Conclusione e Sviluppi Futuri

In uno *sviluppo futuro* questo sistema potrà agevolare il lavoro delle aziende municipalizzate preposte alla raccolta dei rifiuti in una *Smart City*, migliorando il servizio offerto ed evitando di lasciare inutilizzabili alcuni dei contenitori dislocati per le strade.

Questo impianto è realizzabile mediante **LoRa**, protocollo di telecomunicazioni senza fili che permette di coprire distanze dell'ordine del Km. L'architettura del sistema a valle del server, invece, avrà un funzionamento identico a quello mostrato in precedenza.





de Candia Giuseppe  
Fasciano Corrado  
Porcelli Gianluca  
Zippo Paola